VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

WEBOVÁ APLIKACE REDAKČNÍHO SYSTÉMU PRO SPRÁVU DOKUMENTŮ

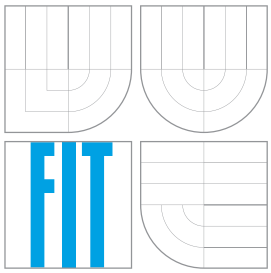
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MARTIN POHNER

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

WEBOVÁ APLIKACE REDAKČNÍHO SYSTÉMU PRO SPRÁVU DOKUMENTŮ

WEB APPLICATION FOR DOCUMENT MANAGEMENT SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MARTIN POHNER

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL DROZD

BRNO 2013

Abstrakt

Tato diplomová práce se zabývá vývojem pokročilé webové aplikace. Účelem této aplikace je rozšířená správa docx dokumentů zahrnující import, editaci, uchovávání a tvorbu nových dokumentů. Vývoj zahrnuje analýzu a stanovení požadavků na aplikaci, studium technologie Java Enterprise Edition a standardu Office Open XML pro formát docx. Dále se zabývá návrhem, implementací a testováním aplikace.

Abstract

The focus of this diploma thesis is a development of advanced web application for. The purpose of the application is extended .docx files management including import, editing, storage and creating new documents. The development comprises analysis, setting the requirements for the application, analysis of the Java Enterprise Edition and Office Open XML standard for the .docx file format. The thesis further deals with design, implementation and testing the application.

Klíčová slova

Webová aplikace, syntaktický analyzátor, docx dokument, zabezpečení, Java Enterprise Edition, Java Server Faces, Hibernate.

Keywords

Web application, parser, docx document, security, Java Enterprise Edition, Java Server Faces, Hibernate.

Citace

Martin Pohner: Webová aplikace redakčního systému pro správu dokumentů, diplomová práce, Brno, FIT VUT v Brně, 2013

Webová aplikace redakčního systému pro správu dokumentů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Michala Drozda.

.....
Martin Pohner
22. května 2013

Poděkování

Děkuji panu Ing. Michalu Drozdovi, za vedení, konzultace a pomoc při práci.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Moderní doba klade důraz na zabezpečení, výkonnost a udržovatelnost webových aplikací, což vedlo ke vzniku a následnému vývoji pokročilých vícevrstvých technologií. Standardní platforma Javy dala vzniku čtyřvrstvé technologii Java Enterprise Edition, která přinesla nový přístup pro tvorbu robustních webových aplikací. Základní jednotky pro stavbu výsledných aplikací tvoří webové komponenty a komponenty řídící logiku aplikace nebo manipulaci s databází. Aplikační server zahrnuje kontejnery, které spravují komponenty a poskytují jim svou funkcionalitu.

Možnosti pokročilých webových aplikací nachází své uplatnění v oblastech, které vyžadují již zmíněné požadavky včetně spolehlivosti. Hitem bankovní sféry je příchod aplikací poskytující kvalitní služby internetového bankovnictví, tzn. provádění složitých transakcí, které musí splňovat danou míru spolehlivosti a bezpečnosti z důvodu důvěry zákazníka (bezhotovostní bankovní převody). Mezi další oblasti uplatnění patří rezervační a zákaznické portály. Využití pokročilé technologie doposud nedosáhlo svého vrcholu a je více než pravděpodobné rozšíření do dalších oblastí uplatnění.

Cílem diplomové práce je seznámit se s problematikou implementace webových aplikací v prostředí Java Enterprise Edition a nastudovat dílčí technologie pro vývoj prezentační a enterprise vrstvy. Dále nastudovat strukturu dokumentů splňujících standard Office Open XML a stěžejní část sestává ze studia zpracování těchto dokumentů. Poté je potřeba provést návrh aplikace splňující všechny požadavky – uživatelské prostředí umožňující rychlou práci s aplikací, uživatelsky definovaný syntaktický analyzátor, podle kterého jsou rozpoznávány importované části dokumentů, import, vytváření dokumentů a zabezpečení proti neautorizovanému přístupu. Po provedení návrhu následuje implementace v jazyce Java, ke které lze použít volně dostupné knihovny. Konečnou aplikaci je nutné otestovat z pohledu funkčnosti, bezpečnosti a kolizních stavů syntaktického analyzátoru a na závěr zhodnotit dosažení výsledky a diskutovat možná rozšíření této aplikace.

1.1 Skladba práce

Kapitola 2 se zabývá analýzou a specifikací požadavků. V této etapě jsou definovány požadavky na aplikaci, uživatelské role a funkcionalita aplikace formou diagramu případů užití. Na závěr je provedena analýza zpracovávaného dokumentu, na jejímž základě je vytvořen model vztahů dat.

Struktuře dokumentům docx splňujících standard Office Open XML je věnována kapitola 3. V této kapitole je popsána souborová struktura dokumentu a dále jsou rozebrány

významné elementy. Závěrečná část se věnuje dostupným knihovnám pro práci s docx dokumentu a jsou popsány jejich výhody a nevýhody.

V kapitole 4 je popsána architektura technologie Java Enterprise edition a její jednotlivé produkty (dílčí technologie). Jedná se o technologie pro implementaci prezentační vrstvy (Servlet, Java Server Pages, Java Server Faces a jejich rozšíření) a business vrstvy (objektově-relační frameworky). Dále jsou popsány aplikační servery pro běh aplikací.

V kapitole 5 je popsán návrh webové aplikace pro správu dokumentů, mezi jehož základní části patří návrh uživatelského prostředí, databáze a syntaktického analyzátoru.

Následuje popis implementace aplikace v kapitole 6, která ukazuje postupný vývoj jednotlivých částí. Podrobněji je vysvětlen způsob implementace doménové vrstvy, uživatelského prostředí a syntaktického analyzátoru.

Testování konečné aplikace z pohledu funkčnosti a bezpečnosti je popsáno v kapitole 7. Každý test obsahuje popis cíle, postupu, kterým bude vykonán a zhodnocení dosažených výsledků.

Kapitola 2

Analýza a specifikace požadavků

Analýza a specifikace požadavků, jak již vyplývá z názvu kapitoly, je etapa, která má za úkol shromáždit a strukturovat veškeré dostupné informace o vyvíjené aplikaci. Informace jsou poté vyobrazeny pomocí diagramů a tabulek, které poskytují lepší pohled na daný problém.

Pro získání požadavků jsou využívány různé známe metody – například interview, studium poskytnutých materiálů, dotazník a podobně. V našem případě získáme požadavky ze zadání práce, z interview a z poskytnutých materiálů.

Na výslednou aplikaci je vždy kladeno mnoho různých typů požadavků, a proto byly již dříve tyto požadavky rozčleněny do několika kategorií (funkční, bezpečnostní a další). V našem případě se budeme zabývat pouze požadavky kladenými na funkčnost, bezpečnost a na vývoj aplikace. Základní diagram této etapy je diagram případů užití, který zobrazuje funkční požadavky a aktéry pracující s aplikací. Významné či komplexnější případy užití jsou vhodně doplňovány o detail případu užití, který se dle stanoveného formátu zobrazuje formou tabulky.

V této kapitole je popsán základní popis aplikace a dále jsou specifikovány požadavky na funkčnost, bezpečnost a vývoj aplikace. Požadavky jsou zobrazeny formou diagramu případů užití a u významných případů jsou navíc zobrazeny jejich detaily. V závěru kapitoly je provedeno studium poskytnutého dokumentu, dle kterého je vytvořen model dat formou ER (Entity Relationship) diagramu.

2.1 Popis aplikace

Cílem této práce je vytvořit webovou aplikaci pracující ve vnitřní síti, která umožňuje bezpečnou správu dokumentů typu docx. Dokumenty obsahují auditní informace a každý dokument má ucelený formát. Aplikace tedy umožňuje importování, uchovávání, vyhledávání, editaci a tvorbu nových dokumentů. Před vstupem do aplikace je vyžadováno ověření uživatele a jeho operace ukládání, mazání a editace zaznamenává logovací systém.

Importování dokumentu lze provádět standardně pomocí průvodce importem. Obsah dokumentu je zpracováván na základě konfigurace syntaktického analyzátoru, která je načtena z xml souboru. Zpracovaný obsah je uložen do databáze.

Vyhledávat informace lze pomocí klíčového slova nebo využitím funkce pokročilého vyhledávání, která upřesňuje soubor vyhledávacích dat. Vyhledávání je dostupné z každé stránky aplikace.

Vytváření nových dokumentů je prováděno stylem nakupování v internetovém obchodě. Po dokončení nákupu nálezů a potvrzení obsahu košíku je spuštěn průvodce generování nového dokumentu. Aplikace vygeneruje nový dokument na základě předpřipravené šablony.

V aplikaci pracují dva typy uživatelů. Běžný uživatel využívá standardní funkčnost aplikace (import, vyhledávání atd.). Administrátorský účet vlastní přístup k nastavení aplikace, které tvoří správa uživatelů, správa auditů, logovací systém, konfigurace syntaktického analyzátoru a parametry uploadu.

Uživatelské rozhraní aplikace poskytuje přehlednou a intuitivní práci.

2.2 Specifikace požadavků

Tato specifikace požadavků se zabývá požadavky kladených na funkčnost, bezpečnost a vývoj. Funkční požadavky plynou z popisu aplikace a pro jejich zobrazení se využívá diagram případů užití.

Tato sekce obsahuje definici rolí, na jejímž základě je vytvořen diagram, který zobrazuje chování systému z hlediska uživatelů. Dále jsou popsány požadavky kladené na bezpečnost a vývoj aplikace.

2.2.1 Definice rolí

Aplikace obsahuje dvě uživatelské role:

- Uživatel – importuje nové dokumenty do databáze pomocí průvodce. Dále spravuje dokumenty, kapitoly a nálezy, mazat smí pouze ty části, které vlastní. Vyhledávat může ve všech nálezech umístěných v databázi. Formou nákupního košíku může vybrat části, které chce vygenerovat do nového dokumentu. Uživatel smí měnit své osobní údaje a heslo.
- Administrátor – vede správu uživatelů a auditů. Dále definuje parametry uploadu a konfiguruje syntaktický analyzátor. Pro kontrolu nad aplikací a uživateli má k dispozici seznam logovacích informací. Dědí veškerá práva uživatelů.

2.2.2 Diagram případů užití

Diagram případů užití zachycuje chování aplikace z hlediska uživatele. Každý případ užití charakterizuje určité použití systému účastníkem. Účastník reprezentuje uživatele, nikoli systém. V diagramu jsou zobrazeny dva typy uživatelských rolí – uživatel a administrátor, přičemž administrátor dědí veškeré případy užití od běžného uživatele. Diagram případů užití je zobrazen na obrázku 2.1.

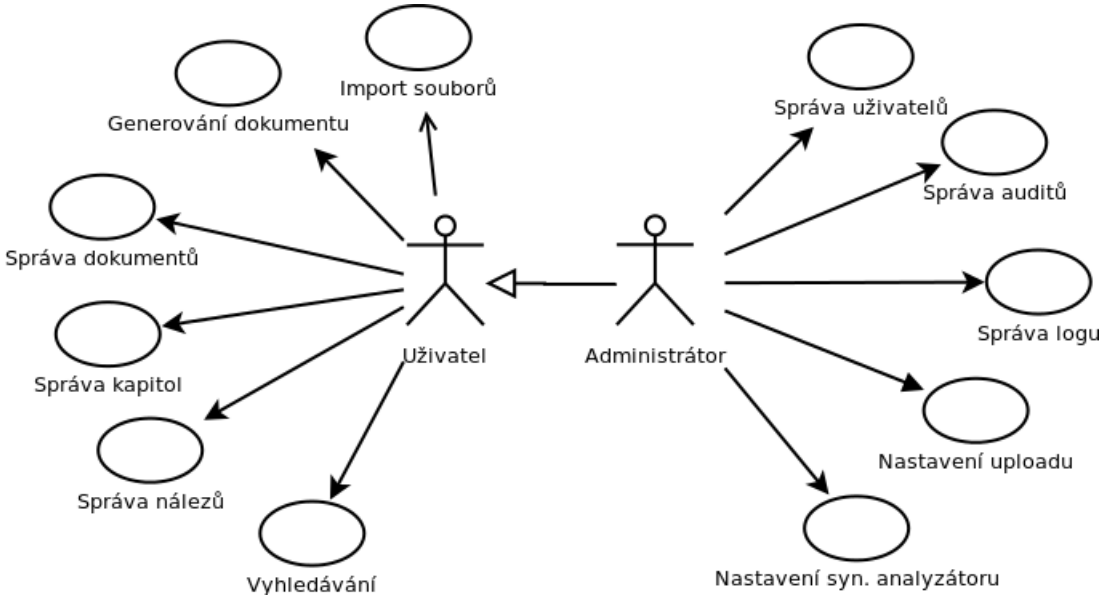
Případy užití uživatele:

- Import Souborů – případ charakterizuje výběr souborů a následný import dat pomocí průvodce.
- Generování dokumentu – případ charakterizuje výběr nálezů formou košíku a následné generování nového dokumentu pomocí průvodce.
- Vyhledávání – případ charakterizuje standardní a rozšířené vyhledávání v nálezech.

- Správa dokumentů – případ charakterizuje dílčí případy CRUD[1].
- Správa kapitol – případ charakterizuje dílčí případy CRUD.
- Správa nálezů – případ charakterizuje dílčí případy CRUD.

Případy užití administrátora:

- Správa uživatelů – případ charakterizuje dílčí případy CRUD.
- Správa auditů – případ charakterizuje dílčí případy CRUD.
- Nastavení uploadu – případ charakterizuje konfiguraci nastavení uploadu.
- Nastavení syntaktického analyzátoru – případ charakterizuje konfiguraci syntaktického analyzátoru.
- Správa logu – případ charakterizuje dílčí případy RD.



Obrázek 2.1: Diagram případů užití.

2.2.3 Případ užití „Import souborů“

Detail případu užití je zobrazen v přehledné tabulce 2.1 určené k tomuto účelu.

2.2.4 Případ užití „Generování dokumentu“

Detail případu užití je zobrazen v přehledné tabulce 2.1 určené k tomuto účelu

[1] Případy CRUD označují případy C – Vytvoření, R – Čtení, U – Editace, D – Smazaní.

Identifikátor	UA01		
Název	Import souborů		
Popis	Uživatel importuje nové dokumenty do databáze.		
Priorita	1 = vysoká	**Frekvence**	Často
Vstupní podmínky	1. Uživatel musí být přihlášen do systému. 2. V databázi musí existovat alespoň jeden audit.		
Výstupní podmínky	1. Systém naimportuje dokument do databáze.		
Uživatelé	Běžný uživatel, Administrátor		
Základní posloupnost	1. Systém zobrazí stránku pro výběr souborů. 2. Uživatel zvolí soubory pro import a potvrdí jejich zpracování. 3. Systém zpracuje importovaný dokument na základě konfigurace syntaktického analyzátoru a zobrazí základní informace z titulní strany a strom kapitol dokumentu. 4. Uživatel zkontroluje titulní stranu dokumentu, nastaví jazyk, ve kterém je dokument napsán a ze stromu kapitol zvolí části pro importování. U kapitol první úrovně zvolí typ auditu. 5. Uživatel potvrdí importování dokumentu do databáze. 6. Systém zobrazí rekapitulaci naimportovaných části a možnost přejít na krok 3 v případě, že je ve frontě další soubor.		
Alternativní posloupnost	2. Uživatel nevybere soubory, zvolí soubor nepodporovaného formátu, velikost souboru převyšuje maximální povolenou velikost. (a) Systém zobrazí varovné hlášení popisující daný problém. 3. Systém nezpracuje importovaný dokument. (a) Systém zobrazí varovné hlášení popisující daný problém. 5. Uživatel nezvolil typ auditu a jazyk dokumentu. (a) Systém zobrazí varovné hlášení popisující daný problém.		

Tabulka 2.1: Detail případu užití Import souborů.

Identifikátor	UA02		
Název	Generování dokumentu		
Popis	Uživatel vloží nálezy do košíku a pomocí průvodce generováním vygeneruje nový dokument.		
Priorita	1 = vysoká	**Frekvence**	Často
Vstupní podmínky	1. Uživatel musí být přihlášen do systému. 2. V databázi musí existovat alespoň jeden nález.		
Výstupní podmínky	1. Systém vygeneruje nový dokument.		
Uživatelé	Běžný uživatel, Administrátor		
Základní posloupnost	1. Systém zobrazí seznam nálezů. 2. Uživatel vloží nálezy do košíku a přejde na stránku s obsahem košíku. 3. Uživatel v případě potřeby poupraví obsah vložených nálezů a spustí průvodce. 4. Uživatel vyplní základní informace o dokumentu a počet kapitol, do kterých chce vložit nálezy. 5. Uživatel vyplní názvy kapitol. 6. Uživatel zařadí nálezy do kapitol. 7. Uživatel v rekapitulaci zkontroluje informace o dokumentu a provede akci generovat dokument. 8. Systém vygeneruje dokument.		
Alternativní posloupnost	2. Uživatel nevložil nález do košíku. (a) Systém zobrazí varovné hlášení popisující daný problém. 4.,5. Uživatel nevyplnil některou z požadovaných položek. (a) Systém zobrazí varovné hlášení popisující daný problém. 8. Systém nevygeneroval dokument. (a) Systém zobrazí varovné hlášení popisující daný problém.		

Tabulka 2.2: Detail případu užití Generování dokumentu.

2.2.5 Požadavky na bezpečnost

Aplikace musí zajistit čtyři hlediska bezpečnosti:

- důvěrnost – k datům mají přístup pouze autorizovaní uživatelé,
- integrita – data smí modifikovat pouze autorizovaní uživatelé,
- dostupnost – musí být zajištěn přístup autorizovanému uživateli,
- nepopiratelnost – operace uživatele jsou zaznamenávány (odpovědnost).

Pro splnění hledisek bezpečnosti je nutné zvolit technologii s bezpečnou architekturou. Bezpečný aplikační server musí poskytovat:

- přístup k chráněným datům musí zajišťovat transportní vrstva HTTPS (Hypertext Transfer Protocol Secure),
- ověření klienta na základě certifikátu.

2.2.6 Požadavky na vývoj

Na vývoj aplikace jsou kladeny následující technické požadavky:

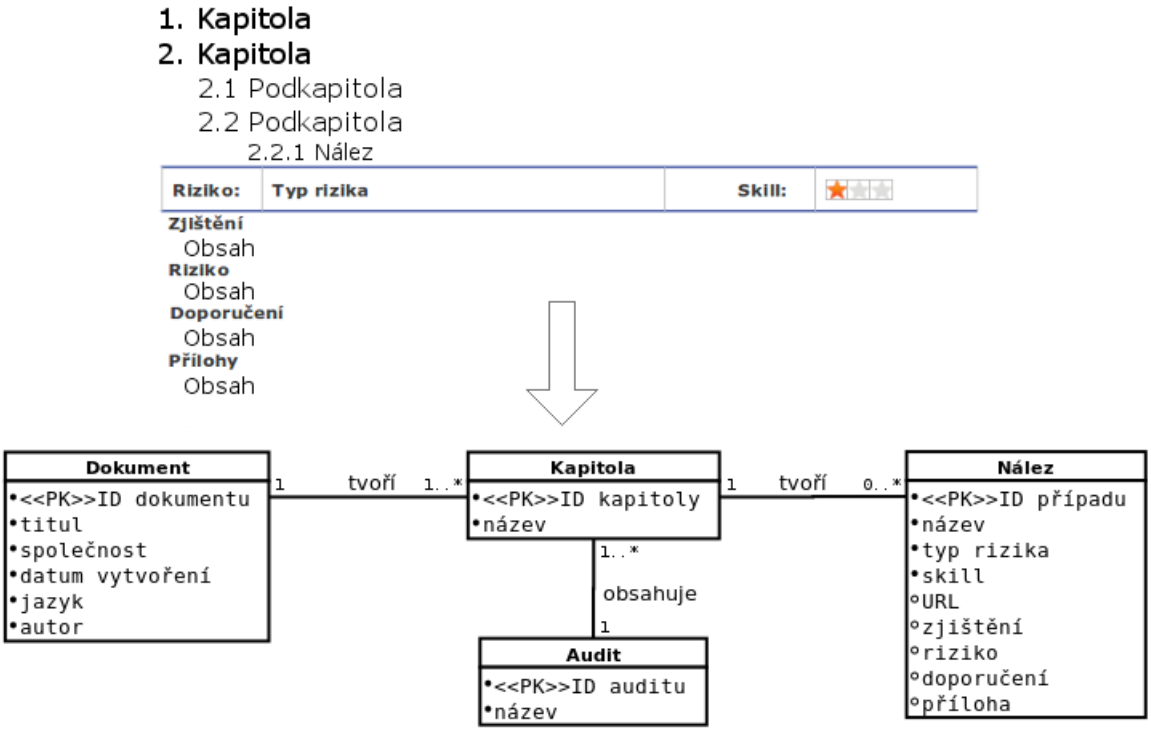
- udržovatelnost budoucí aplikace,
- licence vývojových nástrojů – open source,
- bezpečnost – bezpečná architektura technologie,
- podpora docx – podpora pro zpracování formátu souborů docx.

2.3 Model dat

Pro vytvoření modelu dat bylo provedeno studium dokumentu obsahujícího anonymizovaná auditní data. Jedná se o soubor Microsoft Word s příponou docx.

Pro naši aplikaci jsou důležité pouze některé části, a to základní informace o dokumentu (titul, společnost, datum vytvoření, jazyk a autor), kapitoly první úrovně, kde každá kapitola obsahuje typ auditu, a jednotlivé nálezy. Každý nález pak sestává z názvu, typu rizika, skill, url, zjištění, rizika, doporučení a přílohy. Formát členění dokumentu je naznačen v horní části obrázku 2.2.

Výsledný model dat, jehož koncept je na obrázku 2.2 níže, byl vytvořen pomocí ER diagramu. Entitní množiny modelují dokument, kapitolu, audit a nález. Atributy entitních množin u nichž je uvedeno plné kolečko, jsou nezbytnou součástí nálezu. Na základě tohoto diagramu bude později vytvořeno relační schéma databáze.



Obrázek 2.2: Formát dokumentu a odpovídající ER diagram.

Kapitola 3

Zpracovávaný dokument

Zpracování dokumentů Microsoft Word s příponou docx je jeden z hlavních požadavků na vyvíjenou aplikaci. Typ těchto dokumentů se poprvé objevil s verzí Microsoft Office 2007 a byl definován specifikací Office Open XML pro souborový formát na ukládání dokumentů kancelářských balíků jako textových dokumentů. Finální standard vydalo sdružení ECMA pod číslem ECMA-376 1st edition. Nyní již existuje čtvrté vydání tohoto standardu a je zpětně kompatibilní s předchozími verzemi. Tato kapitola čerpá z prvního vydání [1].

Pro zpracování a vytváření docx dokumentů pomocí programovacího jazyka Java existují volně dostupné knihovny, které poskytují funkce jako například převod dokumentu do XHTML (Extensible HyperText Markup Language) nebo do formátu PDF. Dále umožňují například měnit obsah existujících dokumentů.

V této kapitole je popsány struktura dokumentu docx a dostupné knihovny pro práci s docx dokumenty.

3.1 Dokument docx

Dokument docx je archiv ZIP reprezentovaný sérií souvisejících částí. Pro získání dílčích částí dokumentu je nutné soubor přejmenovat na archiv s příponou zip a poté jej rozbalit vhodným archivačním nástrojem.

V této sekci je popsána vnitřní souborová strukturu dokumentu docx a dále struktura hlavního dokumentu s popisem vybraných elementů.

3.1.1 Souborová struktura

Hierarchii souborů tvoří složky obsahující dokumenty formátu xml. Struktura je zobrazena v tabulce 3.1.

3.1.2 Základní struktura dokumentu

Základní strukturu jednoduchého wordového dokumentu tvoří následující elementy:

- document – Kořenový element definující základní dokument.
- body – Kontejner obsahující strukturu bloků, které zahrnuje hlavní dokument.
- p – Odstavec.

Soubory	Popis
docProps/app.xml	Základní informace o dokumentu, počet slov, odstavců atd.
docProps/core.xml	Informace o autorovi a poslední modifikaci.
_rels/.rels	Vztahy pro propojení základních částí dokumentu.
word/media	Složka s obrázky, které jsou součástí dokumentu.
word/_rels	Vztahy pro propojení dílčích součástí.
word/theme/theme1.xml	Motiv sady Office.
word/style.xml	Definice stylů dokumentu (nadpisy, odstavce).
word/settiings.xml	Nastavení dokumentu, např. zoom, poslední změny.
word/document.xml	Definice formátu a obsahu těla dokumentu.
word/...	Další konfigurační soubory a soubory definující části dokumentu.
[ContentTypes].xml	Konfigurační soubor obsahující strukturu typů obsahu (cesta k obsahu a typ obsahu).

Tabulka 3.1: Struktura dokumentu docx.

- r – Region.
- t – Text.
- tbl – Tabulka.
- pict – Element, který může obsahovat např. textové pole.
- drawing – Obrázek.

Odstavec je kolekce regionů. Region je speciální úsek textu definující dodatečné vlastnosti. Text musí být spojen s alespoň jedním regionem. Struktura dokumentu je znázorněna v kódu 3.1.

```
<w:document ...>
  <w:body>
    <w:p>
      <w:r>
        </w:t>Struktura</w:t>
      </w:r>
    </w:p>
    <w:tbl> ... </w:tbl>
    <w:pict> ... </w:pict>
    ...
  </w:body>
</w:document>
```

Kód 3.1: Formát dokumentu.

Strukturu dokumentu mohou dále tvořit kromě odstavců také např. subdokumenty (kapitoly) <w:subDoc>. Je nutné zdůraznit, že části, jako například. záhlaví, zápatí nebo poznámky pod čarou tento dokument neobsahuje. Formát těchto částí je definován v samostatných souborech.

3.1.3 Odstavce a regiony

Odstavec definuje odlišný úsek textu začínající novým řádkem. Jedná se o základní jednotku definovanou elementem <w:p>. Odstavec může obsahovat tři vnořené elementy – vlastnosti odstavce <w:pPr>, region s obsahem <w:r> a souhrn revizí <w:rsid>. Každý region specifikuje formátování vlastního obsahu a odstavec specifikuje pravidla pro celý odstavec (zarovnání, ohraničení, řádkování a další). Region kromě textu může obsahovat i obrázek <w:drawing>. Pro příklad je uveden zápis kódu 3.2, který definuje odstavec se zarovnáním na střed, kurzívu a obsah „Slovo“.

```
<w:p>
  <w:pPr>                          <!-- vlastnosti odstavce -->
    <w:jc w:val="center"/>
    <w:rPr>                        <!-- vlastnosti každého
       regionu -->
      <w:i/>                       <!-- vlastnost kurzíva -->
    </w:rPr>
  </w:pPr>
  <w:r>
    <w:rPr>                        <!-- vlastnosti regionu -->
      <w:i/>                       <!-- vlastnost kurzíva -->
    </w:rPr>
    <w:t>Slovo</w:t>
  </w:r>i
</w:p>
```

Kód 3.2: Zápis odstavce.

3.1.4 Tabulky

Tabulka je definována základním elementem <w:tbl> a může obsahovat tři vnořené elementy – vlastnosti tabulky <w:tblPr>, rozvržení sloupců <w:tblGrid> a řádek tabulky <w:tr>. Řádek tabulky dále obsahuje vlastnosti řádku <w:trPr> a jednotlivé buňky <w:tc>. Každá buňka může obsahovat vlastní nastavení <w:tcPr> a obsah <w:p>. Pro příklad je uveden zápis kódu 3.2, který definuje obecnou strukturu tabulky.

```
<w:tbl>
  <w:tblPr>
  <!-- Nastavní okrajů, zarovnání, šířka tabulky -->
  </w:tblPr>
  <w:tblGrid>
  <!-- Nastavení šírky sloupců -->
  </w:tblGrid>
  <w:tr>
    <w:trPr>
    <!-- Nastavní vlastností řádku -->
    </w:trPr>
    <w:tc>
      <w:tcPr>
      <!-- Nastavení vlastností buňky-->
```

```
      </w:tcPr>
      <w:p>Obsah buňky</w:p>
    </w:tc>
  </w:tr>
</w:tbl>
```

Kód 3.3: Zápis tabulky.

3.1.5 Styly dokumentu

Styly definují vizuální stránku dokumentu, jedná se o předdefinovaný set odstavců, znaků a nadpisů, který se aplikuje na text dokumentu.

Zápis 3.4 zobrazuje příklad formátu souboru se styly. Zobrazený styl definuje odstavec s identifikátorem a názvem Heading1. Intuitivně lze chápat, že se jedná o nadpis první úrovně s velikostí písma 22.

```
<w:styles>
  <w:style w:type="paragraph" w:styleId="Heading1">
    <w:name w:val="heading1"/>
    <w:next w:val="Normal"/>
    <w:link w:val="Heading1Char"/>
    <w:rPr>
      <w:sz w:val="22"/>
    </w:rPr>
    ...
  </w:style>
  ...
</w:styles>
```

Kód 3.4: Styly dokumentu.

3.1.6 Vztahy pro propojení dílčích součástí

Vztahy slouží pro propojení souborové struktury a dalších částí dokumentu. Vztah reprezentuje identifikátor, typ vztahu a cíl, který obsahuje například cestu k souboru či hypertextový odkaz. Kód 3.5 zobrazuje příklad formátu souboru se vztahy.

```
<Relationships>
  <Relationship Id="rId1"
                Type="http://schemas.openxmlformats.org/
                   officeDocument/2006/relationships/image"
                Target="media/image6.png"/>
</Relationships>
```

Kód 3.5: Vztahy dokumentu.

3.2 Konfigurace syntaktického analyzátoru

Z modelu dat 2.3 a analýzy zpracovávaného dokumentu 3.1.2 lze odvodit pravidla pro syntaktický analyzátor, který z dokumentu vyčlení požadované části. Pro rozeznání textových

polí na titulní straně je nutné pořadí jednotlivých polí a vnitřních odstavců. Pro rozeznání typu kapitoly, podkapitoly či nálezu je nutný identifikátor stylu. Pro rozeznání rizika nálezu je potřeba seznam všech rizik v požadovaných jazycích. Pro přiřazení obrázků rizik a skill je nutný seznam všech obrázků. Pro rozeznání části nálezu je nutný seznam všech nadpisů v požadovaných jazycích. Konfiguraci syntaktického analyzátoru zobrazuje kód 3.6.

```
<?xml version="1.0" encoding="UTF-8">
<rules>
  <text-boxes>
    <text-box type="title_client" position="1">
      <p type="title" position="1" />
      <p type="client" position="4" />
    </text-box>
    <text-box type="date_authors" position="2">
      <p type="date" position="1" />
      <p type="authors" position="4" />
    </text-box>
  </text-boxes>
  <images>
    <risk-images>
      <img name="info">info.png</img>
      ...
    </risk-images>
    <skill-images>
      <img name="ko" value="0">ko.png</img>
      ...
    </skill-images>
  </images>
  <styles>
    <style type="h1">id-nadpis-1</style>
    <style type="h2">id-nadpis-2</style>
    ...
    <style type="text">id-text</style>
  </styles>
  <finding-parts>
    <part name="url">
      <cs>URL</cs>
      <en>Application</en>
    </part>
    ...
  </finding-parts>
  <risks>
    <risk name="info" value="0">
      <cs>Informativní</cs>
      <en>Info</en>
    </risk>
    ...
  </risks>
```

```
</rules>
```

Kód 3.6: Konfigurace syn. analyzátoru.

3.3 Knihovny pro práci s docx dokumenty

Pro práci s docx dokumenty existují dvě významné knihovny Docx4java a Javadocx. Následuje popis knihoven z hlediska práce s docx dokumenty. Knihovny jsou implementovány v jazyce Java a jsou volně dostupné k použití.

3.3.1 Docx4java

Knihovna Docx4java poskytuje práci s docx, pptx a xslx dokumenty založenou na JAXB (Java Architecture for XML Binding) a podporuje standard ECMA-376 1st edition, tedy dokumenty vytvořené v nástroji Microsoft Office 2007. S dokumenty od verze 2010 se mohou vyskytnout problémy.

Mezi významné funkce této knihovny patří transformace dokumentu docx na dokument formátu XHTML. Knihovna při převodu dbá na vysokou vizuální podobu výstupního XHTML dokumentu, což zpomaluje samotný převod.

Další významnou funkcí je vytvoření nového dokumentu pomocí elementů či transformace XHTML na dokument docx. Knihovnu je možné stáhnout na webové stránce [7], kde se nachází i dokumentace.

3.3.2 Javadocx

Volně dostupná verze knihovny Javadocx poskytuje omezenou práci s dokumenty docx a podporuje standard ECMA-376 1st edition.

Převod dokumentu do XHTML probíhá v rychlejším čase než u knihovny Docx4java, a její výstup se již moc nepodobá jednotlivým částem dokumentu.

Vytváření nových dokumentů lze pouze pomocí elementů. Knihovnu je možné stáhnout na webové stránce [12], kde se nachází i dokumentace.

Kapitola 4

Java Enterprise Edition

Java Enterprise Edition (Java EE) je rozšíření základní platformy Java Standard Edition (Java SE) o možnosti vývoje vícevrstvých serverových aplikací, pro jejichž běh je nutný aplikační server. Tato technologie se v současnosti velmi často využívá pro budování aplikací např. internetové bankovnictví a některé zákaznické portály. Tato kapitola převážně čerpá z oficiální dokumentace, která je dostupná z [5].

Architektura technologie Java EE je zobrazena na obrázku 4.1 a tvoří ji tyto čtyři vrstvy:

- Klientská – poskytuje přístup k aplikačnímu serveru pomocí např. webového prohlížeče nebo klientské aplikace.
- Webová – prostředník mezi klientským webovým prohlížečem a business logikou. Jedná se o základní nezbytnou logiku, která zajišťuje generování obsahu (X)HTML a udržování stavu sezení. Vrstva může být realizována pomocí servletů, JSP, JSF a JavaBeans.
- Business – realizuje hlavní funkcionalitu aplikace např. pomocí Enterprise JavaBeans (EJB), Java Persistacne API (JPA). EJB tvoří dva typy – session beans a message-driven beans. JPA provádí objektově-relační mapování objektů na tabulky relační databáze.
- Enterprise information – poskytuje spojení s databází a dotazování. Realizace pomocí JDBC (Java Database Connectivity), Java Persistence API, Java Transaction Architecture (JTA). JDBC slouží pro volání SQL příkazů a dotazů na databázi.

Tato kapitola představuje aplikační servery a dílčí technologie jednotlivých vrstev Java EE.

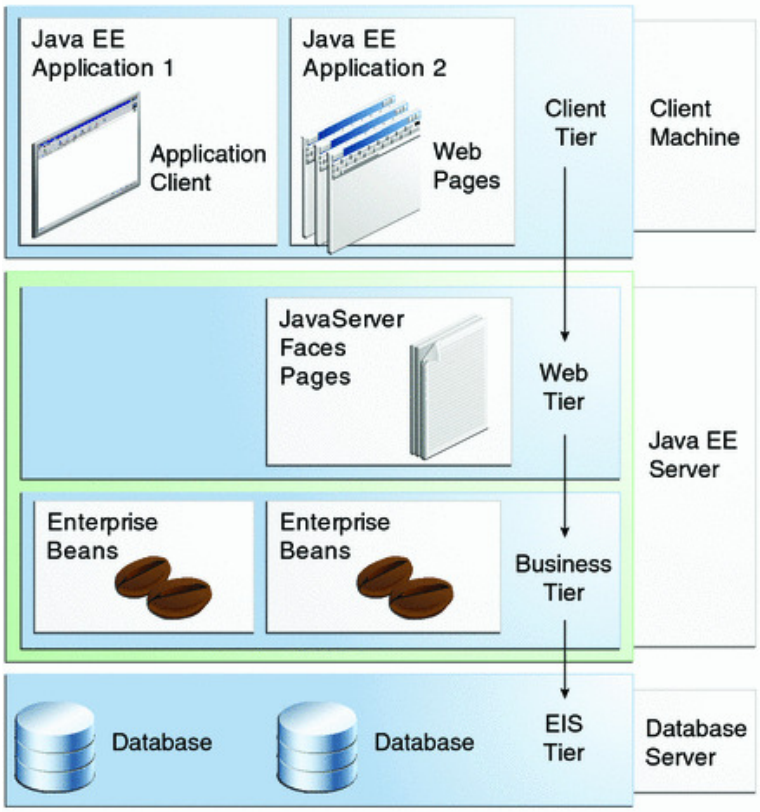
4.1 Aplikační servery

Aplikační server je prostředí, které obsahuje řadu rozhraní, kontejnerů pro provoz enterprise aplikací. Mezi významné funkce je možné zařadit např. mapování uživatelských rolí, nastavení SSL, ověření pomocí certifikátu, podpora transakcí atd.

Tato sekce obsahuje popis vybraných, volně dostupných aplikačních serverů.

4.1.1 Apache TomEE

Apache TomEE vychází ze základní verze Apache Tomcat, která pouze vykonává Java servlety a konvertuje JSP stránky na servlety 4.2.1. Apache TomEE je rozšířen o knihovny



Obrázek 4.1: Architektura Java EE převzato z [5].

podporující práci s EE vrstvou. Server je možné stáhnout z webové stránky [11], kde se nachází i dokumentace.

4.1.2 GlassFish

Aplikační server GlassFish je webový server společnosti Oracle plně podporující platformu Java EE. GlassFish podléhá licenci GPL a CDDL, jehož hlavní účel je ukázka implementace nových rysů platformy Java EE. Komerční verze je nazvána Oracle GlassFish Server.

Architektura serveru je založená na modulárním jádře poháněném OSGi (Open Service Gateway initiative) systémem, který umožňuje vzdálenou instalaci, odinstalaci, aktualizaci aplikací a komponent bez nutnosti restartu serveru. Server je možné stáhnout z webové stránky [6], kde se nachází i dokumentace.

4.2 API webové vrstvy

Webová vrstva (webový kontejner) zajišťuje základní logiku aplikace a obsahuje celou řadu aplikačních rozhraní (kontejnerů). Tato sekce obsahuje popis rozhraní Java Servlet a Java Server Pages. Rozhraním Java Server Faces a Java Persistence jsou věnovány samostatné sekce.

4.2.1 Java Servlet

Servlet je speciální třída napsaná v jazyce Java, která rozšiřuje schopnosti serveru přistupovat k hostitelským aplikacím prostřednictvím programovacího modelu dotaz-odpověď. Servlety mohou odpovědět na jakýkoli typ dotazu a jsou užity pro rozšíření aplikací umístěných na webovém serveru. Pro takové aplikace Java Servlet definuje HTTP-specific servlet třídy.

Pro psaní servletů jsou k dispozici balíky `javax.servlet` a `javax.servlet.http` poskytující rozhraní a třídy. Každý servlet musí implementovat rozhraní `Servlet`, které definuje metody životního cyklu. Třída `HttpServlet` poskytuje metody `doGet` a `doPost` pro řízení HTTP požadavků.

Životní cyklus servletu je řízen kontejnerem, ve kterém je umístěn. Kontejner při požadavku na servlet vykoná následující kroky:

1. Pokud instance servletu neexistuje, kontejner zavede servlet třídu a vytvoří její novou instanci. Provede inicializaci voláním metody `init`.

2. Vyvolá metodu `service` procházející objekty dotaz a odpověď.

4.2.2 Java Server Pages

Java Server Pages (JSP) je nadstavba nad Java Servlet a umožňuje snadno vytvářet webové aplikace se statickými a dynamickými komponentami. Pro tuto technologii jsou dostupné všechny možnosti Java Servletů. Hlavní rysy JSP tvoří:

- Jazyk pro vývoj JSP stránek (textové dokumenty popisující zpracování požadavku a odpovědi).

- Výrazový jazyk pro přístup k serverovým objektům.

- Mechanismus pro definování rozšíření JSP jazyka.

JSP stránka je obyčejný textový dokument obsahující dva typy textu – statická data, která mohou být vyjádřena pomocí např. HTML, XML a JSP elementy, které sestavují dynamický obsah. Soubory JSP mají příponu .jsp.

JSP elementy mohou být vyjádřeny dvěma způsoby syntaxe, standardní a XML. Pro jeden soubor je jeden typ syntaxe. JSP konstrukce tvoří čtyři základní skupiny – komentáře, direktivy, skriptovací elementy a akce. Při požadavku webového prohlížeče na určitou JSP stránku se provede následující postup:

1. Interpretuje se JSP stránka.

2. Vygeneruje se Java servlet.

3. Servlet se pomocí standardního překladače převede do bajtového kódu.

4. Servlet je načten do virtuálního stroje Java.

5. Dojde k vyvolání služební metody servletu a odeslání odpovědi.

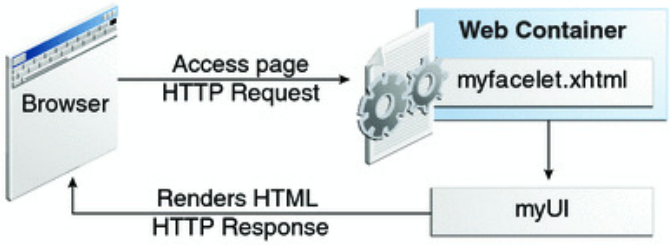
4.3 Java Server Faces

Java Server Faces (JSF) je framework, který poskytuje sadu komponent pro budování webových aplikací na straně serveru. Framework JSF je nástupcem JSP a tvoří API pro reprezentaci komponent a řízení jejich stavu, řízení událostí, datovou konverzi, definici navigace stránky, podporu přístupnosti.

JSF zde zavádí nový deklarativní jazyk Facelets, který se využívá pro vytváření vizuální podoby JSF stránek spolu s jazykem HTML. Hlavní rysy Facelets tvoří užití XHTML pro vytváření stránek, podpora Facelets knihovny elementů, podpora výrazového jazyka a šablonování pro komponenty a stránky. Základní JSF aplikaci tvoří:

- Set webových stránek bez komponent.
- Výrazový jazyk.
- Set tagů (značek), které přidávají komponenty do webových stránek.
- Set Managed Beans – řídicí kontroléry.
- Web.xml soubor popisující nastavení aplikace.
- Faces-config.xml soubor definující navigaci aplikace, uživatelské objekty a komponenty.
- Set uživatelských tagů (značek) reprezentující uživatelské objekty.

Obrázek 4.2 reprezentuje architekturu aplikace využívající technologii JSF. Webová stránka myfacelet.xhtml je složena z tagů vkládajících komponenty do výsledného zobrazení myUI (reprezentace webové stránky na straně serveru). Výsledná stránka je odeslána jako odpověď klientské aplikaci.



Obrázek 4.2: Architektura JSF aplikace převzato z [5].

Základní sadou objektů JSF je knihovna Mojara, pro kterou existuje celá řada rozšíření uživatelských objektů (komponent), které jsou postavené na AJAX (Asynchronous JavaScript and XML) a podporují HTML5. Každý objekt je specifikován XML kódem a při vytváření HTTP odpovědi se na základě tohoto kódu generuje výsledný XHTML kód. Díky těmto komponentám lze vybudovat profesionální aplikace.

Tato sekce dále obsahuje popis vybraných knihoven uživatelských objektů.

4.3.1 Ice Faces

Knihovna Ice Faces je open-source projekt s licencí Apache 2, která umožňuje další vývoj bez jakýchkoli komerčních omezení. Pomocí technologie Automatic Ajax je zde zajištěna

minimální obnova stránek. Knihovna poskytuje bohatý set JSF komponent, ze kterých lze sestavit moderní a bohaté uživatelské prostředí. Vývojové prostředí Netbeans obsahuje integraci této knihovny, která je volně dostupná z webové stránky [3], kde se nachází i dokumentace.

4.3.2 Prime Faces

Knihovna Prime Faces je stejně jako Ice Faces open-source projekt a poskytuje set komponent pro budování uživatelských prostředí (např. HtmlEditor, Dialogová okna, grafy). Knihovna je nezávislá a nevyžaduje jakoukoli konfiguraci. Pro komponenty je připraveno třicet druhů grafických témat a je možné vytvářet vlastní témata pomocí webového nástroje. Vývojové prostředí Netbeans obsahuje integraci této knihovny, která je volně dostupná z webové stránky [8], kde se nachází i dokumentace.

4.4 Java Persistence

Java Persistence API (JPA) je specifikace, která poskytuje objektově-relační mapování pro správu relačních dat v aplikacích. JPA je složeno ze čtyř oblastí, samotné JPA, dotazovací jazyk, Java Persistence Criteria API a objektově-relační mapování metadat.

Entitní třídy reprezentují tabulky relační databáze. Každá instance této třídy koresponduje s příslušným řádkem tabulky. Požadavky na entitní třídu:

- Třída musí být anotována `javax.pesistence.Entity` anotací.
- Třída musí mít public nebo protected konstruktor bez argumentů.
- Třída nesmí deklarovat `final`.
- Třída musí implementovat rozhraní `Serializable`, pokud bude instance třídy pracovat se session bean.
- Třída může dědit z entitní i ne-entitní třídy.
- Instanční proměnné musí být deklarovány jako private či protected.

Aby bylo možné uložit entitní třídu do databáze, musí obsahovat atributy pouze daných typů (Primitivní typy, řetězce, Serializable typy, výčtové typy, kolekce).

Pro implementaci JPA existují různé frameworky např. EclipseLink, Hibernate či OpenJPA. Tato sekce dále popisuje vybranou implementaci Hibernate.

4.4.1 Hibernate

Hibernate je rozsáhlý framework pro práci s persistentními daty a poskytuje knihovny pro fulltextové vyhledávání (Hibernate Search). V souvislosti s JPA specifikací lze tento framework použít jako implementaci. Hibernate je dostupné z webové stránky [9], kde se nachází podrobná dokumentace.

4.5 Enterprise JavaBeans

Business vrstva (kontejner) EJB zajišťuje hlavní logiku aplikace. Serverové komponenty Enterprise beans mohou pracovat samostatně nebo mohou komunikovat s jinými komponentami a vykonávat business logiku na serveru. Aplikace je zde členěna hierarchicky mezi více procesů/vláken. Enterprise JavaBeans poskytuje dva typy beanů, Session Bean a Message-driven bean.

4.5.1 Session bean

Session Bean je komponenta umožňující vykonat určitou práci pro klienta, který si to vyžádá (synchronní komunikace). Jedná se o objekt, který obsahuje aplikační logiku nějakého business procesu. Instanci této komponenty je možné použít vícekrát pro více klientů, ale maximálně pro jednoho klienta současně. Session bean může provádět např. databázové operace a existují tři typy session beanů:

- Stateless – Bean slouží pro obsloužení konverzace, která sestává z jednoho volání a není třeba uchovávat nějaký konverzační stav mezi voláními více metod.
- Statefull – Bean uchovává konverzační stav po celou dobu sezení s jedním klientem. Vhodné pro udržování stavu nákupního košíku v internetovém obchodě.
- Singleton – Bean je instanciován pouze jednou a existuje po celou dobu životnosti aplikace.

4.5.2 Message-driven bean

Message-driven bean reprezentuje v business vrstvě Java Message Service (JMS), který umožňuje asynchronní reakce na události (přijímání zpráv). Zprávy mohou být zaslány jakoukoli Java EE komponentou (webová komponenta, jiný bean). Tento typ beanu se liší oproti Session beanu v tom, že klient nepřistupuje k message-driven benu pomocí rozhraní. V případě, že nechceme používat synchroní blokování přijetí v komponentě na straně serveru, využijeme message-driven bean.

Kapitola 5

Návrh aplikace

Tato kapitola obsahuje návrh webové aplikace pro správu auditních dat, kde bude možné importovat jednotlivé dokumenty docx do databáze a generovat nové dokumenty. V rámci návrhu aplikace bylo nutné vyřešit následující:

- uživatelské prostředí,
- filtrování protokolu HTTP,
- syntaktický analyzátor,
- generátor nových dokumentů,
- vyhledávání v databází a
- zabezpečení aplikace.

5.1 Použité technologie a knihovny

Aplikační server byl zvolen GlassFish, který standardně podporuje práci s Enterprise Java Beans a disponuje webovým uživatelským rozhraním. Aplikace je navržena tak, aby byla zprovoznitelná i na aplikačním serveru TomEE.

Technologie aplikačního rozhraní byla zvolena Java Server Faces, která je nástupcem Java Server Pages a odděluje prezentační vrstvu od logiky aplikace. Pro rozšíření sady základních uživatelských objektů JSF byla zvolena knihovna Prime Faces, která poskytuje dokonalejší komponenty než Ice Faces. Jako implementace JPA byl zvolen framework Hibernate, který obsahuje na rozdíl od ostatních dostupných frameworků i fulltextové vyhledávání, které je potřebné pro vyhledávání v databázi.

Pro práci s dokumenty docx byly zvoleny obě uvedené knihovny Docx4java a Javadocx. První zmíněná knihovna bude využita pro generování nového dokumentu, protože poskytuje funkci zpětného převodu z formátu XHTML na dokument docx. knihovna JavaDocx poslouží pro zpracování importovaného dokumentu.

Databázové prostředí bylo zvoleno MySQL z důvodu dobré orientace v tomto prostředí. Další použité knihovny a jejich funkci lze vidět v tabulce 5.1:

Všechny využité knihovny jsou volně dostupné k použití.

Knihovna	Funkce
Jsoup	práce s DOM modelem dokumentu XHTML
PrettyURL	čisté URL adresy

Tabulka 5.1: Další použité knihovny a jejich funkce.

5.2 Uživatelské prostředí

Webová uživatelská prostředí mohou být díky současným technologiím velice různorodá se spoustou grafických efektů. Uživatelské prostředí bylo navrženo tak, aby splňovalo standardní webové rozvržení a jeho síla byla výhradně v práci s aplikací, srozumitelnosti a jednoduchosti. Navrhnout inovativní řešení pro uživatelské prostředí není cílem této práce.

Uživatelské prostředí naší aplikace tvoří přihlašovací stránka a souhrn zabezpečených stránek (jádro aplikace). Uživatel přistupuje k prostředí pomocí webového prohlížeče, který zobrazuje výsledek odpovědi webového serveru. Jednotlivé webové stránky obsahují odkazy, tlačítka, boxy, textová pole, nadpisy a další komponenty.

Tato sekce popisuje návrh jazykové lokalizace aplikace, rozvržení stránky, šablony stránky a klientských stránek. Dále je popsán návrh navigace stránek, čistých URL adres a grafického prostředí.

5.2.1 Jazyková lokalizace

Jazyková lokalizace uživatelského prostředí je navržena tak, aby podporovala vícejazyčnost (aplikace nastaví lokalizaci dle lokalizace webového prohlížeče).

Lokalizace pro jednotlivé jazyky je uložena v samostatných souborech s názvem tvaru `msg_cz.properties`, které obsahují vždy dvojice klíč-text. Nutný přídavek `cz` v názvu souboru za podtržítkem udává typ lokalizace, na jehož základě aplikace rozlišuje tyto soubory. Strukturu lokalizačního souboru zobrazuje kód 5.1.

```
user=Uživatel
pass=Heslo
```

Kód 5.1: Struktura lokalizačního souboru.

V konfiguračním souboru `faces-config.xml` je nutné nastavit lokalizaci aplikace, aby byl umožněn přístup k lokalizačním souborům. Nastavení lokalizace obsahuje defaultní jazyk lokalizace a připojení lokalizačních souborů. Strukturu nastavení zobrazuje kód 5.2.

```
<application>
  <resource-bundle>
    <base-name>resources.msg</base-name>
    <var>msg</var>
  </resource-bundle>
</application>
```

Kód 5.2: Struktura nastavení lokalizace.

Anglickou verzi lokalizace prozatím aplikace podporovat nebude, protože to není v rozsahu mého zadání, ale pro další rozvoj aplikace jsou podklady připraveny. Jedná se zejména o přeložení české lokalizace do angličtiny a úpravy lokalizačního nastavení.

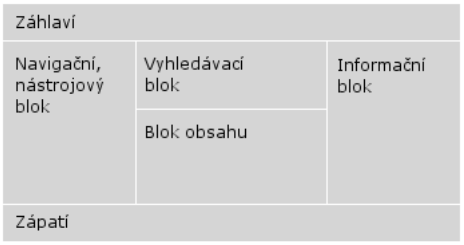
5.2.2 Rozvržení stránky

Klasické rozvržení webových aplikací spočívá v rozdělení okna prohlížeče do horizontálních a vertikálních bloků, které jasně definují typ informace, kterou zobrazují. Účelem rozvržení je zvýšení orientace uživatele v aplikaci.

Rozvržení stránky navrhované aplikace je zobrazeno na obrázku 5.1 a rozčleněno do šesti základních bloků:

- Záhlaví – název aplikace.
- Navigační, nástrojový blok – administrátorské a nástrojové menu, které umožňuje pracovat s aplikací.
- Vyhledávací blok – vyhledávání dat v databázi.
- Blok obsahu – jediný dynamický blok, jehož obsah se mění při vyhledávání a zobrazení obsahu jednotlivých stránek.
- Informační blok – nákupní košík, informace o přihlášeném uživateli, poslední přidané dokumenty.
- Zápatí – umístění ochranné známky, autora a možné navigační prvky.

Díky tomuto statickému rozvržení bude práce s aplikací rychlejší a srozumitelnější.



Obrázek 5.1: Rozvržení uživatelského prostředí.

5.2.3 Šablona stránky

Šablona stránky je speciální XHTML stránka obsahující statické a dynamické části dle rozvržení na obrázku 5.1. Mezi další dynamické části lze zařadit titulek stránky.

Dynamické bloky je nutné v šabloně stránky označit speciálním elementem `<ui:insert></ui:insert>`, který bude nahrazen vloženým obsahem. V naší šabloně stránky je nutné označit místa pro vkládání dynamického titulku a dynamického obsahu. Kód 5.3 zobrazuje zjednodušený návrh šablony stránek.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets">
  <h:head>
    <title><ui:insert name="pageTitle"></ui:insert></title>
  </h:head>
  <h:body>
    <!-- záhlaví stránky -->
```

```
    <!-- navigační blok -->
    <!-- vyhledávací blok -->
    <ui:insert name="content"></ui:insert>
    <!-- informační blok -->
    <!-- zápatí stránky -->
  </h:body>
</html>
```

Kód 5.3: Návrh šablony stránek

Na obrázku 5.2 je zobrazena grafická šablona stránky odpovídající přesně definovanému rozvržení z 5.2.2. V navigačním bloku se nachází nástrojové a administrátorské menu a v informačním bloku jsou zobrazeny stav košíku (počet položek), aktuální přihlášený uživatel a seznam nejnovějších přidaných dokumentů.

Obrázek 5.2: Grafický návrh šablony stránky aplikace.

5.2.4 Klientská stránka

Klientská stránka je speciální XHTML stránka, která obsahuje pojmenované elementy, jejichž obsah bude vložen na definovaná místa v šabloně stránky. Kód 5.4 zobrazuje zjednodušený návrh klientské stránky.

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
                xmlns:ui="http://java.sun.com/jsf/facelets"
                template="./template.xhtml">
  <ui:define name="pageTitle">
    Název titulku stránky
  </ui:define>
  <ui:define name="content">
    Obsah stránky
  </ui:define>
```

```
</ui:composition>
```

Kód 5.4: Návrh klientské stránky.

5.2.5 Navigace stránek

JSF (Prime Faces) komponenty, mohou vyvolat přesměrování na základě hodnoty atributu `action`, která může obsahovat následující hodnoty:

- přesměrovávací řetězec,
- odkaz na metodu řídícího beanu, která provede přímé přesměrování nebo vrací přesměrovávací řetězec.

Přímé přesměrování lze provést pomocí objektu reprezentující kontext aplikace. V naší aplikaci byla zvolena přehlednější možnost přesměrování pomocí řetězců. Tento princip řízení navigace stránek vyžaduje definici navigačních pravidel v konfiguračním souboru `faces-config.xml`. Každé navigační pravidlo obsahuje identifikátor stránky, ze které je provedeno přesměrování, a seznam případů. Každý případ obsahuje přesměrovávací řetězec a identifikátor výsledné stránky. Struktura navigačního pravidla je znázorněna kódem 5.5, který provede přesměrování z jakékoli stránky na stránku index.xhtml.

```
<navigation-rule>
  <from-view-id>*</from-view-id>
  <!-- Symbol * značí libovolnou stránku -->
  <navigation-case>
    <from-outcome>home</from-outcome>
    <to-view-id>/index.xhtml</to-view-id>
    <redirect />
  </navigation-case>
</navigation-rule>
```

Kód 5.5: Struktura navigačního pravidla.

5.2.6 Třídy řídící uživatelské prostředí

Třídy řídící uživatelské prostředí (Managed Beans) jsou speciální třídy, jejichž metody lze volat z šablony stránek a z klientských stránek. Tyto třídy využívají anotace @ManagedBean (označuje řídící třídu) a @SessionScoped(označuje zařazení těchto tříd do objektů sezení). Dále tyto třídy komunikují se Session beans a vytváří obsah klientských stránek.

5.2.7 Čisté URL adresy

Dnešní webové aplikace běžně využívají čisté URL adresy místo standardních identifikátorů stránek zejména kvůli SEO analýze. V našem případě se jedná spíše jen o vizuální efekt.

Knihovna Pretty URL upravuje HTTP požadavky na základě mapovacího filtru a pravidel. Mapovací filtr je umístěn v konfiguračním souboru `web.xml` a jeho struktura popsána v části 5.3. Mapovací pravidla jsou umístěna v konfiguračním souboru `pretty-config.xml` a každé pravidlo obsahuje mapovací identifikátor, mapovací řetězec a původní identifikátor stránky. Strukturu mapovacího pravidla zobrazuje kód 5.6, který nahrazuje název stránky login.xhtml za řetězec login.

```
<url-mapping id="login">
  <pattern value="/login" />
  <view-id value="/faces/login.xhtml" />
</url-mapping>
```

Kód 5.6: Struktura mapovacího pravidla.

5.2.8 Grafický vzhled prostředí

Grafický vzhled prostředí tvoří kaskádové styly, které definují zobrazení základních HTML elementů, základních JSF uživatelských komponent a rozšířených Prime Faces komponent.

Téma vzhledu pro komponenty Prime Faces bylo navrženo pomocí webového návrháře témat na oficiálních stránkách, který vytvoří pro všechny komponenty definice kaskádových stylů. Přístup k vygenerovanému tématu je nutné nastavit v konfiguračním souboru `web.xml`.

Grafický vzhled šablony stránky, který je uveden na obrázku 5.2, byl navržen v nástroji GIMP.

5.2.9 Přihlašovací stránka

Pro vstup do aplikace slouží jednoduchá webová stránka obsahující formulář pro vyplnění přihlašovacích údajů. Po třetím neúspěšném přihlášení musí uživatel navíc ještě vyplnit pole Captcha. Tato stránka není šablonou stránek a nejedná se ani o klientskou stránku. Přihlašování je zobrazeno na obrázku 5.3.

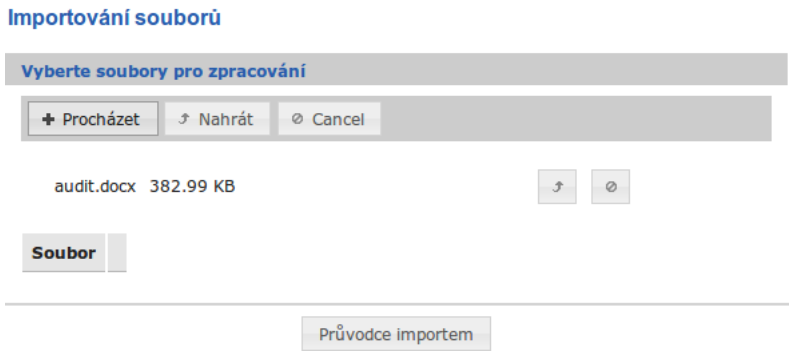
(a) Přihlášení. (b) Třetí neúspěšné přihlášení.

Obrázek 5.3: Přihlášení do aplikace.

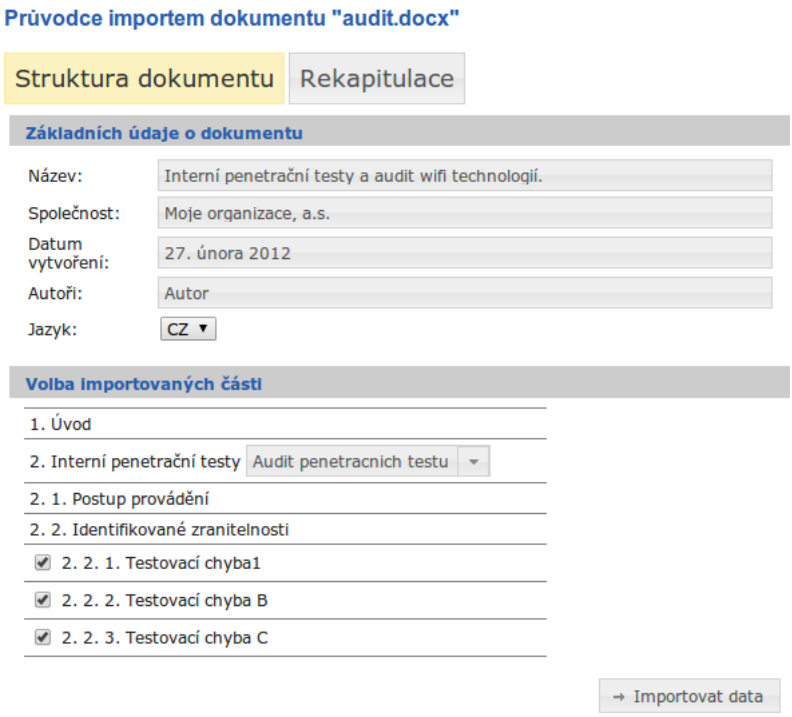
5.2.10 Vybrané případy užití

Pro ukázku uživatelského prostředí charakterizující daný případ užití jsou zde uvedeny jednotlivé obrazovky aplikace. Jedná se výhradně o blok obsahu z šablony stránek.

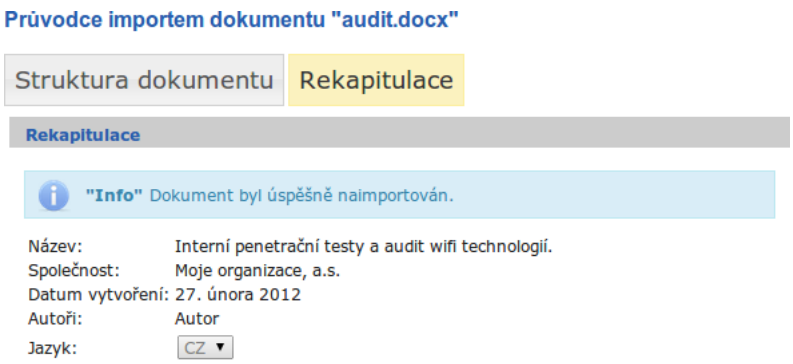
Případu užití „Import souborů“ postupně odpovídají obrázky 5.4, 5.5 a 5.6. Obrázek 5.4 zobrazuje první krok procedury importování souboru. Uživatel zde vybere dokumenty, které odešle na server. Na obrázku 5.5 uživatel zkontroluje základní údaje o dokumentu, nastaví jazyk dokumentu, zvolí typ auditu druhé kapitoly a vybere nálezy, které chce importovat. Na obrázku 5.6 je k dispozici stručná rekapitulace proběhlého importu. Kroky průvodce se postupně opakují pro každý dokument.



Obrázek 5.4: Výběr a umístění dokumentů na server.



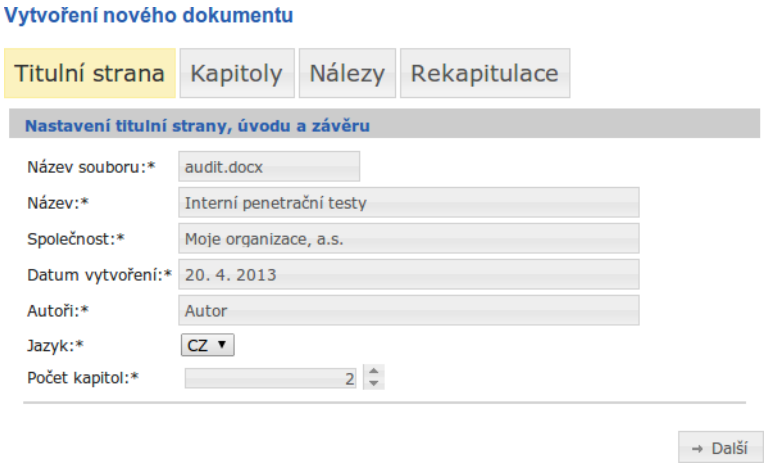
Obrázek 5.5: Potvrzení informací o dokumentu, volba auditu druhé kapitoly a výběr importovaných částí.



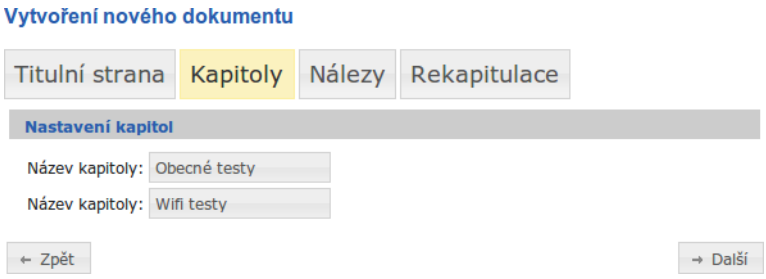
Obrázek 5.6: Rekapitulace provedeného importu.

Případu užití „Generování dokumentu“ postupně odpovídají obrázky 5.7, 5.8, 5.9, 5.10 a 5.11. Obrázek 5.7 zobrazuje obsah nákupního košíku, kde je možné smazat, upravit jednotlivé položky či vyprázdnit celý košík. Obrázek 5.8 zobrazuje první krok průvodce generováním dokumentu. Uživatel zde nastaví informace, které budou doplněny na budoucí titulní stranu výsledného dokumentu, zvolí počet kapitol a nastaví jazyk. Obrázek 5.9 znázorňuje nastavení názvů kapitol. Obrázek 5.10 zobrazuje možnost seřazení kapitol a jednotlivých nálezů. Přesouvat položky lze stylem drag-drop, nebo pomocí levého postranního menu. Obrázek 5.11 zobrazuje stručnou rekapitulaci před vygenerováním nového dokumentu.

Obrázek 5.7: Obsah košíku s nálezy.



Obrázek 5.8: Nastavení titulní strany dokumentu a počtu kapitol.



Obrázek 5.9: Nastavení názvů kapitol.

Obrázek 5.10: Seřazení obsahu dokumentu.

Obrázek 5.11: Rekapitulace před vygenerováním dokumentu.

5.3 Filtrování protokolu HTTP

Filtrování poskytuje správu nad HTTP zprávami a lze jej využít například k omezení přístupu na zabezpečenou stránku. Předtím než je zpráva předána Java Servletu, a může byt pozměněna.

Tato sekce obsahuje základní strukturu filtru a dále návrh jednotlivých filtrů, které budou využity v aplikaci.

5.3.1 Obecná struktura filtru

Filtr je umístěn v konfiguračním souboru `web.xml` a obsahuje identifikátor a třídu, která zpracovává HTTP zprávy. Další nutnou částí je nastavení mapovacích pravidel, kde je určeno jaké typy URL a HTTP zpráv budou zachytávány a kterým filtrem. Základní strukturu filtru a mapování zobrazuje kód 5.7, který filtruje veškeré URL adresy a HTTP požadavky.

```
<filter>
  <filter-name>Filter name</filter-name>
  <filter-class>filter.FilterClass</filter-class>
</filter>
<filter-mapping>
  <filter-name>Filter name</filter-name>
  <url-pattern>/*</url-pattern>
  <dispatcher>REQUEST</dispatcher>
</filter-mapping>
```

Kód 5.7: Struktura filtru.

Pořadí definovaných filtrů v konfiguračním souboru určuje pořadí aplikace jednotlivých filtrů.

5.3.2 Filtr pro kódování znaků

Filtr pro kódování znaků zajišťuje správné zobrazení obsahu HTTP zprávy v aplikaci. Databáze aplikace je v porovnání UTF-8 a webový prohlížeč ne vždy zasílá HTTP zprávy v tomto kódování. V mém případě zasílal zprávy s kódováním ISO-8859-2 a docházelo k chybnému zobrazení českých znaků. Při aplikaci filtru dojde k přenastavení kódování zprávy na UTF-8.

5.3.3 Filtr pro upload souborů

Filtr pro upload souborů obsahuje navíc nastavení maximální velikosti nahraného souboru a cestu k cílovému temp adresáři. Každý nahraný soubor je temporárním souborem, který je po čase opět smazán. Tento filtr vyžaduje komponenta `fileupload` z knihovny Prime Faces.

5.3.4 Filtr pro čisté url

Filtr pro čisté URL zajišťuje filtrování HTTP zpráv, kde dochází k přenastavení čisté URL adresy na identifikátor stránky na základě mapovacích pravidel popsaných v 5.6. Tento filtr vyžaduje knihovna Pretty URL.

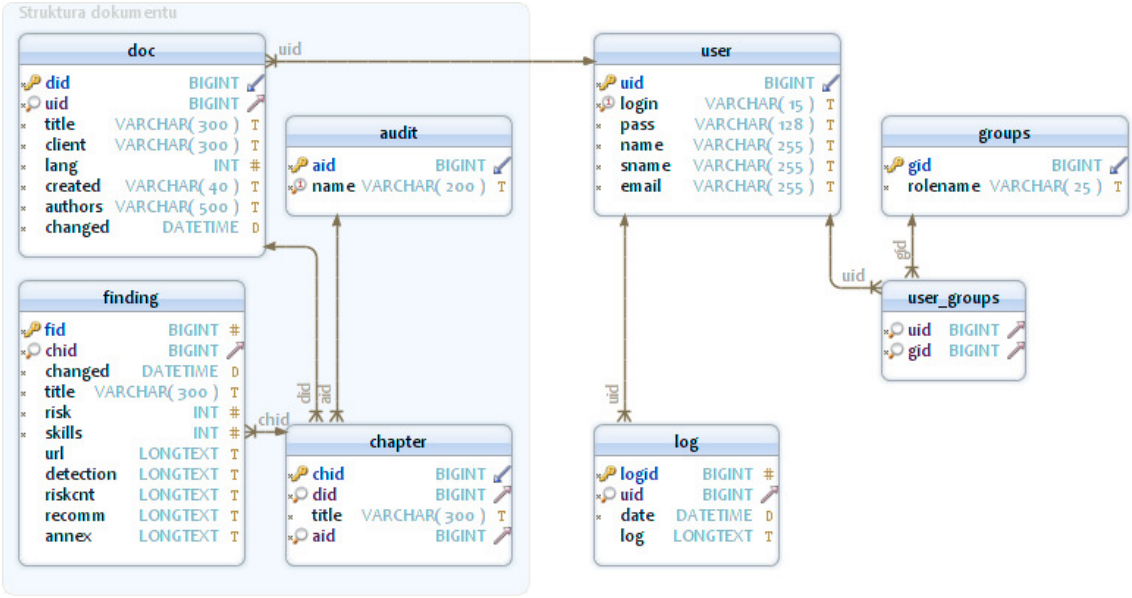
5.3.5 Zabezpečovací filtry

Zabezpečovací filtry omezují přístup k chráněnému obsahu. Při každém požadavku na zabezpečenou stránku je provedeno ověření sezení přihlášeného uživatele. Dále tyto filtry brání k přístupu běžných uživatelů k zabezpečenému obsahu, který je určen výhradně pro administrátora. Zobrazení zdrojového kódu facelets je také zabezpečeno těmito filtry.

5.4 Databáze

Návrh databáze je vytvořen na základě ER diagramu z kapitoly 2.3. Pro převod ER diagramu na schéma relační databáze byla použita známa pravidla viz. [4]. Databáze je umístěna na MySQL databázovém serveru.

Obrázek 5.12 zobrazuje návrh schématu relační databáze. Tabulky `doc`, `chapter` a `finding` představují strukturu dokumentu docx. Tabulka `audit` obsahuje typy auditů přiřazených k dokumentům. Informace o uživateli a jeho uživatelské roli jsou uloženy v tabulce `user`, `user_groups` a `groups`. Logovací data jsou umístěna v tabulce `logs`.



Obrázek 5.12: Relační schéma databáze.

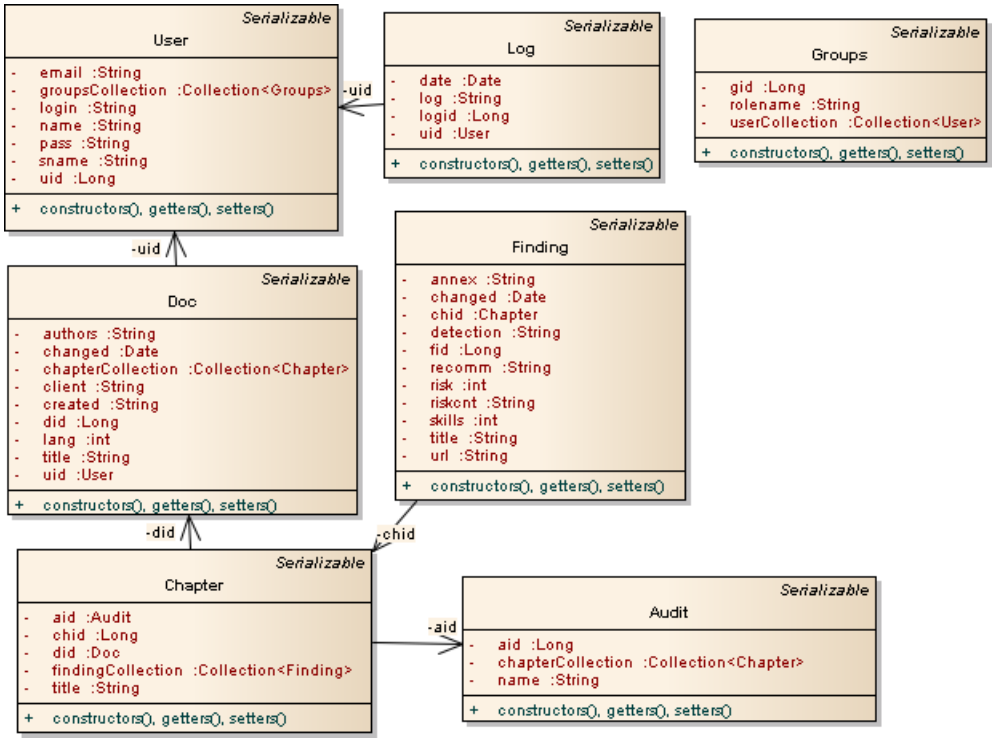
5.5 Diagram tříd doménové vrstvy

Použité entitní třídy a vztahy mezi nimi jsou zobrazeny v diagramu tříd 5.13. Z důvodu velikosti kompletního diagramu je zde uveden pouze zmenšený diagram, který neobsahuje všechny atributy a metody, pouze ty důležitější z pohledu funkce jednotlivých tříd.

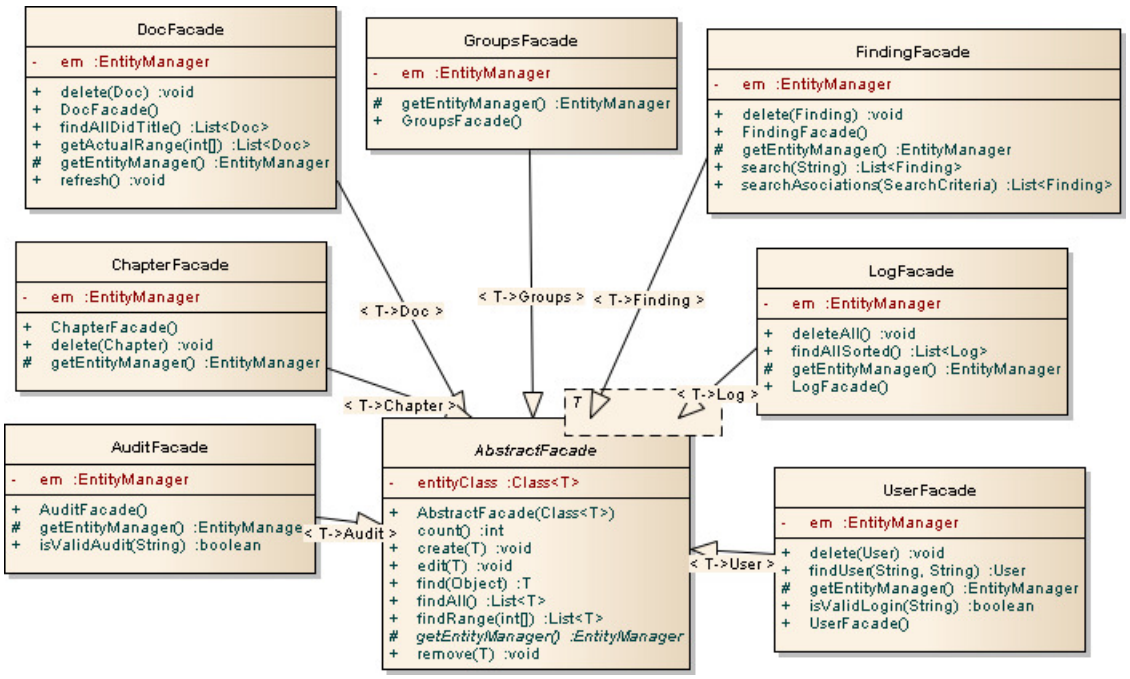
Dále tato sekce obsahuje diagram Session beans, tedy tříd, které pracují s persistentními objekty (s databází).

5.5.1 Diagram Session beans

Použité Session beans a vztahy mezi nimi jsou zobrazeny v diagramu tříd 5.14. Zde je uveden kompletní diagram.



Obrázek 5.13: Zjednodušený diagram entitních tříd.



Obrázek 5.14: Diagram session beans.

5.6 Syntaktický analyzátor

Zpracovávaný dokument docx obsahuje titulní stranu a obsah s auditními daty. Naše aplikace z tohoto dokumentu zpracovává pouze části zobrazené v modelu dat 2.3. Pro práci s dokumenty docx jsou použity knihovny JavaDocx, Jsoup a standardní knihovny Java SE.

Zpracování dokumentu docx lze rozdělit na dvě části:

1. Zpracování titulní strany - informace typu název, klient, datum a autoři.
2. Zpracování obsahu pomocí syntaktického analyzátoru - informace typu obsah, kapitola, nález.

Tato sekce obsahuje návrh zpracování titulní strany a zpracování obsahu pomocí syntaktického analyzátoru.

5.6.1 Zpracování titulní strany

Informace na titulní straně dokumentu jsou umístěny v plovoucích textových polích. Data jsou zpracovány přímo ze zdrojových souborů dokumentu na základě konfigurace pro titulní stranu viz. 3.2.

Postup zpracování titulní strany:

1. Načtení souboru word/document.xml z vnitřní souborové struktury dokumentu docx a převod na DOM (Document Object Model).
2. Z výběru všech elementů `w:txtxContent` jsou vybrány elementy dle pozice určené v konfiguraci.
3. Z každého elementu `w:txtxContent` je proveden výběr elementů `w:p` dle pozice určené v konfiguraci. Uvnitř těchto elementů se již nachází požadovaná informace.

Vlastnosti zpracování:

- Pokud se na daných pozicích nevyskytují požadované elementy, nastane chyba.
- Pokud konfigurace titulní strany neodpovídá zpracovávanému dokumentu je možné, že budou zpracovány chybné informace.

5.6.2 Zpracování obsahu dokumentu

Dokument docx je pomocí knihovny Javadocx převeden na dokument XHTML a dále zpracováván syntaktickým analyzátorem.

Syntaktický analyzátor je stavový automat, který na základě konfigurace z 3.2 zpracovává elementy XHTML dokumentu a vytváří nový strukturovaný XHTML dokument. Syntaktický analyzátor je nezbytný, neboť z převedeného dokumentu nejsou snadno rozpoznatelné souvislosti mezi daty (např. konec kapitoly, nálezu).

Algoritmus 5.1 demonstruje práci syntaktického analyzátoru, jehož vstupem je objektový model dokumentu XHTML a výstupem strukturovaný objektový model dokumentu XHTML. Algoritmus provede načtení konfigurace a dále ze vstupního dokumentu vybere všechny elementy, jejichž třída kaskádového stylu odpovídá kapitole. Poté prochází všechny elementy mezi jednotlivými kapitolami. Na základě zpracování těla každého známého elementu je vytvořena nová reprezentace tohoto elementu, která je vložena do výstupního

dokumentu. Výstupní dokument již obsahuje pouze části odpovídající stanovenému modelu dat 2.3. Kód 5.8 zobrazuje strukturu výstupního dokumentu.

Vlastnosti zpracování:

- Elementy s neznámou třídou kaskádového stylu jsou ignorovány.
- Elementy, které mají chybně označenou třídu např. nález, který je označen třídou nadpis3 (správně nadpis3find), bude zpracován jako nadpis3.
- Obrázky vyskytující se v nálezech jsou z dokumentu zkopírovány.
- Speciální HTML znaky vyjadřující znaky s českou diakritikou jsou převedeny.

Algoritmus 5.1 Algoritmus syntaktického analyzátoru pro zpracování obsahu.

Vstup: document - DOM XHTML dokument
Výstup: structureDocument - DOM XHTML dokument

```
1: loadParserSettings()
2: chapters = document.getElementsByCssClass("h1")
3: for i = 0 to chapters.size() do
4:    elem = chapters.get(i)
5:    chapter = createChapter(elem)
6:    elem = elem.nextElem()
7:    while elem.cssClass() != "h1" do
8:       if elem.cssClass() = "h2" then
9:          h2 = createH2(elem)
10:         chapter.append(h2)
11:      else if elem.cssClass() = "h3" then
12:         h3 = createH3(elem)
13:         chapter.append(h3)
14:      else if elem.cssClass() = "Nález" then
15:         finding = createFinding(elem)
16:         chapter.append(finding)
17:      end if
18:      elem = elem.nextElem()
19:   end while
20:   structureDocument.append(chapter)
21: end for
```

```
<html>
  <head></head>
  <body>
    <div class="chapter"><h2>Nadpis kapitoly</h2>
      <div class="nadpis2">Nadpis kapitoly 2. úrovně</div>
      <div class="nadpis3">Nadpis kapitoly 3. úrovně</div>
      <div class="nadpis3find">
        <h3>Název nálezu</h3>
        <risk value="Hodnota rizika"/>
        <skills value="Hodnota skill"/>
```

```
        <div class="zjisteni">Obsah zjištění</div>
        ...
      </div>
    </div>
    ...
  </body>
</html>
```

Kód 5.8: Struktura výstupního dokumentu.

5.6.3 Diagram tříd

Použité třídy a vztahy mezi nimi jsou zobrazeny v diagramu tříd 5.15. Z důvodu velikosti kompletního diagramu je zde uveden pouze zmenšený diagram, který neobsahuje všechny atributy a metody, pouze ty důležitější z pohledu funkce jednotlivých tříd.

5.7 Generátor dokumentu

Generátor nového dokumentu využívá předpřipravenou docx šablonu v českém a anglickém jazyce, do které doplní požadované informace. Pro generování dokumentu je použita knihovna Docx4java. Generování probíhá v těchto krocích:

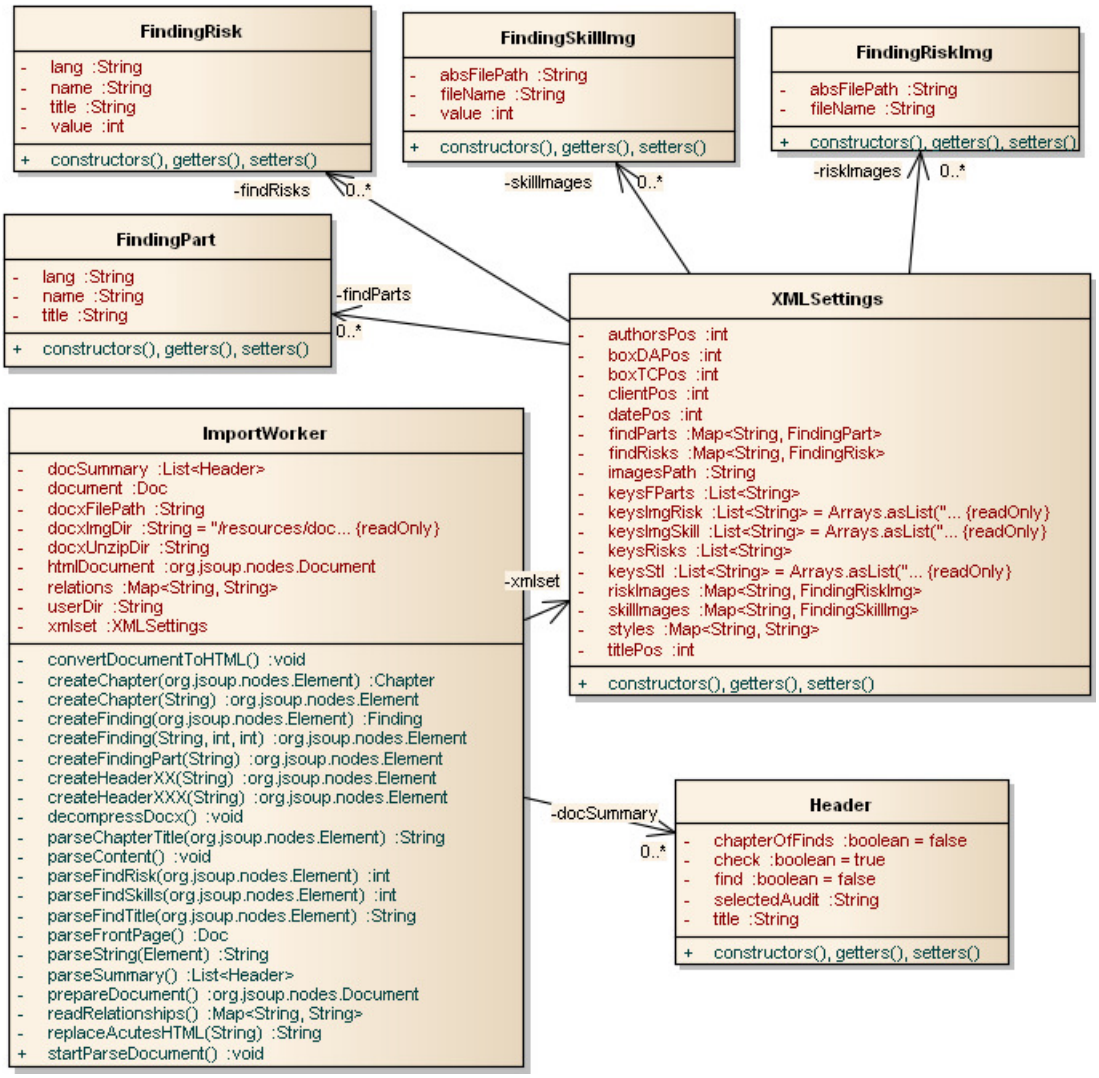
1. Vytvoření kopie šablony.
2. Načtení kopie pomocí knihovny Docx4java.
3. Doplnění údajů titulní strany dokumentu. Dojde k nahrazení předpřipravených klíčových řetězců.
4. Doplnění kapitol a nálezů do šablony. Nadpisy jsou vytvářeny pomocí elementů popsaných v 3.1. Obsah jednotlivých částí nálezů je převeden z formátu XHTML do docx a poté vložen do dokumentu.

5.8 Fulltextové vyhledávání

Pro fulltextové vyhledávání v databázi byl využit framework Hibernate a jeho součást Search, která vyžaduje nastavení cesty k adresáři pro indexování a dále adresářového poskytovatele v konfiguraci persistentní jednotky.

Klasické i rozšířené vyhledávání probíhá v tabulce nálezy (finding), kterou reprezentuje entitní třída `Finding`. Rozšířené vyhledávání navíc omezuje vyhledávací soubor na základě kritérií, která vychází z dalších vlastností entitních tříd `Doc, Chapter, Audit`. Přehled vyhledávacích kritérií je zobrazen v tabulce 5.2.

Fulltextové vyhledávání bude ignorovat slova obsažená v českém a anglickém stop-listě, diakritiku, velikost písmen a html elementy.



Obrázek 5.15: Zjednodušený diagram syntaktického analyzátoru.

Kritérium	**Hodnoty**
Jazyk dokumentu	CZ/EN
Typ auditu	Název auditu
Typ rizika	Info, Low, Medium, High, Critical
Část nálezu	Url, Zjištění, Riziko, Doporučení, Přílohy

Tabulka 5.2: Seznam kritérii pro rozšířené vyhledávání.

5.9 Zabezpečení

Pro přístup k chráněným datům je využita transportní vrstva HTTPS. Obsah je zabezpečen podmínkou řádného přihlášení, kdy po třetím neúspěšném přihlášení je vyžadována Captcha.

Třídy Session beans, které řídí přístup k databázi, využívají chráněných dotazů, na které nelze z formulářů aplikovat útok SQL injection.

Kapitola 6

Implementace

Tato kapitola obsahuje popis implementace aplikace. Jedná se především o řešení tvorby doménové vrstvy, uživatelského prostředí, syntaktického analyzátoru a generátoru nových dokumentů.

Aplikace je implementována v jazyce Java. Základní stavební bloky a základní třídy toho jazyka byly nabyty při studiu a dále v knize [10]. Principy implementace webových aplikací na serveru Glassfish (viz sekce 4.1.2) pomocí technologie Java Server Faces (viz sekce 4.3) byly nabyty v knize [2] a z dostupné internetové dokumentace [5].

6.1 Doménová vrstva

Doménová vrstva je vytvořena pomocí JPA a frameworku Hibernate, které poskytují objektově-relační mapování tříd na tabulky relační databáze. Tato sekce dále popisuje význam persistentní jednotky, vytvoření entitních a session tříd. Nakonec jsou popsány podrobněji metody pro fulltextové a rozšířené fulltextové vyhledávání.

6.1.1 Persistentní jednotka

Persistentní jednotka (Persistence Unit) je konfigurace pro nastavení JPA pro konkrétní sadu entitních tříd a session beans. Nastavení je uloženo v konfiguračním souboru persistence.xml, kde byly nastaveny název, poskytovatel persistentní knihovny (Hibernate) a spojení s databází. Dále zde bylo provedeno nastavení samotného frameworku Hibernate a vyhledávání (Hibernate Search).

6.1.2 Entitní třídy

Pro reprezentaci objektově-relačního modelu byly vytvořeny entitní třídy obsahující speciální mapovací anotace.

Každá deklarace entitní třídy je označena anotací `@entity`, která definuje entitní třídu a `@Table(name = "název tabulky")`, která mapuje modelovanou tabulku. Definici třídy tvoří proměnné (modelují atributy tabulky) a metody, které umožňují práci s proměnnými (`getters` and `setters`) a konstruktory. Následující tabulka 6.1 zobrazuje význam dalších použitých anotací. Pro přehlednost tabulky byly vynechány některé atributy anotací.

Anotace slouží pro zachování konzistence databáze a nedovolí uživateli vytvářet nepovolené vstupy. Při validaci formulářů je provedena kontrola na základě uvedených anotací.

Anotace	Popis
`@Column(name = název sloupce)`	Mapování na sloupec tabulky.
`@GeneratedValue`	Označení primárního klíče.
`@NotNull`	Neprázdnost proměnn.
`@Size(min = 0, max = 255)`	Proměnné typu `String` udává délku řetězce.
`@Temporal(TemporalType.TIMESTAMP)`	Proměnné typu `Date` udává typ datumu.
`@OneToMany(...)`	Vztah entitní třídy a kolekcí daného typu.
`@ManyToOne(...)`	Vztah entitní třídy vůči objektu.
`@ManyToMany(...)`	Vztah mezi entitní třídou a kolekcí daného typu.
`@JoinColumn(...)`	Udává spojení tabulek.

Tabulka 6.1: Seznam významných anotací použitých pro entitní třídy.

Entitní třídy byly vygenerovány pomocí nástroje uvnitř vývojového prostředí Netbeans, který generuje třídy na základě schématu databáze. Pro tabulky reprezentující pouze spojení se samostatné třídy nevytváří.

6.1.3 Session Beans

Třídy business vrstvy (Session Beans) slouží pro práci s persistentními objekty (instance entitních tříd). Každá třída obsahuje anotace `@Stateless`, která udává životní cyklus beanu viz 4.5.1.

Každá definici třídy obsahuje instanci entitního manažeru (Entity Manager), který poskytuje funkcionalitu pro vykonávání operací nad databází. Tato instance je anotována jako `@PersistenceContext(unitName="název persistentní jednotky")`.

Pro kompletní skupinu Session beans byla vytvořena jedna Abstraktní třída, která implementuje základní operace CRUD. V jednotlivých Session beans byly doimplementovány další potřebné metody např. vyhledávání a mazání (delete). Metoda pro mazání uvnitř abstraktní třídy neprováděla v některých případech očekávanou funkci.

Třídy Session beans byly vygenerovány pomocí nástroje uvnitř vývojového prostředí Netbeans, který generuje třídy na základě již vytvořených entitních tříd.

6.1.4 Anotace fulltextového vyhledávání

Entitní třídy `Finding, Doc, Chapter a Audit` byly obohaceny o anotace, jejich název a popis je zobrazen v tabulce 6.2. Pro přehlednost zde nejsou uvedeny atributy některých anotací.

Anotace	Popis
`@Indexed`	Označení entitní třídy, ve které se bude vyhledávat.
`@AnalyzerDef`	Definice analyzátoru vyhledávání.
`@DocumentId`	Označení primárního klíče sloupce pro vyhledávání.
`@Field`	Označení sloupce pro vyhledávání.
`@Analyzer`	Analyzátor sloupce, který aplikuje pravidla během indexování.
`@IndexedEmbedded`	Vestavěný index u vnořených instancí entitních tříd.

Tabulka 6.2: Seznam významných anotací použitých pro vyhledávání.

Kód 6.1 zobrazuje definici analyzátoru s názvem „myanalyzer“, který je důležitou součástí indexování nálezů. Analyzátor obsahuje znakové filtry pro českou diakritiku a pro HTML elementy. Dále obsahuje definované filtry slov pro ignorování velikosti písmen a stop-list. Slova nacházející se v stop-listu nejsou indexována.

```
@AnalyzerDef(name="myanalyzer",
  charFilters = {
    @CharFilterDef(factory = MappingCharFilterFactory.class,
       params = {
      @Parameter(name = "mapping",
        value = "org/mapping-chars.properties") }),
    @CharFilterDef(factory = HTMLStripCharFilterFactory.class)
  },
  tokenizer = @TokenizerDef(factory = StandardTokenizerFactory
     .class),
  filters = {
    @TokenFilterDef(factory = LowerCaseFilterFactory.class),
    @TokenFilterDef(factory = StopFilterFactory.class, params
       = {
      @Parameter(name="words",
        value= "org/stoplist.properties"),
      @Parameter(name="resource_charset", value = "UTF-16BE"),
      @Parameter(name="ignoreCase", value="true") })
})
```

Kód 6.1: Definice analyzátoru.

6.1.5 Fulltextové vyhledávání

Fulltextové vyhledávání provádí metoda `search` umístěná v Session bean `FindingFacade`. Na základě JPA entitního manažeru je vytvořen nový entitní manažer určený pro fulltextové vyhledávání.

Poté je vytvořena instance třídy `QueryParser`, která definuje seznam vyhledávacích atributů tabulky a analyzátor. Tento parser je využit pro zpracování vyhledávacího řetězce. Nakonec je vytvořen klasický JPA dotaz přes rozhraní fulltextu, který obsahuje zpracovaný vyhledávací řetězec a `Finding.class`. Metoda vrací seznam vyhledaných nálezů seřazený dle data poslední změny.

6.1.6 Rozšířené fulltextové vyhledávání

Rozšířené fulltextové vyhledávání provádí metoda `extendedSearch` umístěná v Session bean `FindingFacade`, jejím parametrem je objekt obsahující kritéria vyhledávání viz tabulka 5.2.

Podobně jako u sekce 6.1.5 je zde vytvořen entitní manažer pro fulltextové vyhledávání a nově pak instance třídy `QueryBuilder` pro entitní třídu `Finding`, která slouží pro sestavení složitějšího dotazu s kritérii. Pro různé kombinace zvolených kritérií bylo vytvořeno osm dotazů. Kód 6.2 zobrazuje dotaz, který vyhledá všechny nálezy dle zvoleného jazyka dokumentu, auditu a vyhledávacího řetězce. Podmínka `must` značí, že vyhledaný nález musí odpovídat danému kritériu.

```
query = qb // query builder
 .bool()
   .must( qb.keyword().onField("chid.inDocument_lang").
      matching(langs).createQuery())
   .must( qb.keyword().onField("chid.inAudit_aid").matching(
      audit).createQuery())
   .must( qb.keyword().onField("risk").matching(risks).
      createQuery())
 .createQuery();
```

Kód 6.2: Dotaz pro vyhledávání s kriterii.

Metoda vrací seznam vyhledaných nálezů seřazený dle data poslední změny.

6.2 Uživatelské prostředí

Tato sekce popisuje implementaci uživatelského prostředí, které tvoří XHTML stránky (facelety) a řídící třídy (Managed Beans). XHTML stránky jsou tvořeny komponentami JSF a PrimeFaces a komunikují s řídícími třídami.

6.2.1 Systémová hlášení

Systémová hlášení informují uživatele o stavech operací, které v aplikaci vykonává. Třída `SystemMessage` obsahuje veřejné statické metody pro přidání zprávy do kontextu aplikace. V aplikaci jsou zobrazovány tři typy zpráv:

- informační – typ `SEVERITY_INFO`,
- varování – typ `SEVERITY_WARN`,
- chyba – typ `SEVERITY_ERROR`.

Každá metoda pro přidání zprávy do kontextu aplikace načte lokalizační soubor aplikace, ve kterém jsou umístěna hlášení. Poté je do kontextu `FacesContext` přidána nová zpráva `FacesMessage`, která obsahuje typ a obsah zprávy.

6.2.2 Nastavení syntaktického analyzátoru

Nastavení syntaktického analyzátoru reprezentuje klientská stránka `pars_option.xhtml` a řídící třída `ParsOptionBean`, které spolu komunikují.

Při renderování klientské stránky je načteno XML nastavení do objektu `xmlSet`, které zajišťuje metoda `initialize` řídící třídy. Tato metoda je vyvolána z klientské stránky pomocí elementu `f:event`. Po vyrenderování klientské stránky je zobrazeno menu pomocí elementu `p:tabView`, které umožňuje přepínat mezi jednotlivými skupinami nastavení. Konkrétní typy nastavení objektu `xmlSet` (např. nastavení pozice textového pole pro nadpis titulní strany dokumentu) zobrazují elementy `p:inputText`. Tlačítko pro uložení skupiny nastavení (element `p:commandButton`) je umístěno na každém panelu. Při stisknutí tohoto tlačítka je vyvolána specifická metoda řídící třídy, která zapíše nové nastavení do konfiguračního souboru.

Při požadavku na návrat původního nastavení je vyvolána metoda `actionRestore`, která provede přepsání obsahu konfiguračního souboru defaultním nastavením.

6.2.3 Nastavení uploadu

Nastavení uploadu reprezentuje klientská stránka `upload_option.xhtml` a řídící třída `UserLogin`, které spolu komunikují. Hlavním účelem této třídy je řízení přihlašování uživatelů a uživatelského sezení, kde nastavení uploadu je pouze jednou z jeho částí.

XML nastavení je načteno do objektu `fileUpSet` ihned po přihlášení uživatele do systému a je k dispozici po celou dobu vytvořeného sezení. Přihlašování uživatelů a uživatelského sezení je podrobněji popsáno v 6.2.13.

Po vyrenderování klientské stránky je zobrazeno menu pomocí elementu `p:tabView`. Konkrétní typy nastavení objektu `fileUpSet` (např. nastavení maximální velikosti uploadovaného souboru) zobrazují elementy `p:inputText`. Tlačítko pro uložení skupiny nastavení (element `p:commandButton`), je umístěno na každém panelu. Při stisktnutí toho tlačítka je vyvolána specifická metoda řídící třídy, která zapíše nové nastavení do konfiguračního souboru.

Při požadavku na návrat původního nastavení je vyvolána metoda `actionRestore`, která provede přepsání obsahu konfiguračního souboru defaultním nastavením.

6.2.4 Správa auditů

Správu auditů reprezentují klientské stránky `audit_list.xhtml, audit_edit.xhtml`, `audit_new.xhtml` a řídící třída `AuditBean`.

Během renderování klientské stránky `audit_list` je v řídící třídě načten seznam auditů z databáze. Pro přístup k metodám nad databází slouží instance třídy `AuditFacade` (Session bean).

Seznam auditů v klientské stránce zobrazuje element `h:dataTable` jehož atribut `value` je spojen s metodou řídící třídy vracející seznam objektů (auditů). Tlačítka, vyskytující se na klientské stránce `audit_list`, slouží pro přechod na další již zmíněné klientské stránky a jsou spojeny s metodami řídící třídy. Tyto metody vrací přesměrovávací řetězec dle definovaných navigačních pravidel.

Mezi další metody řídící třídy patří metody pro uložení upravovaného auditu (spojení s klientskou stránkou `audit_edit.xhtml`) a nového auditu (`audit_new.xhtml`) do databáze. Při ukládání nového auditu je provedena kontrola existence názvu auditu.

6.2.5 Správa uživatelů

Správu uživatelů reprezentují klientské stránky `user_list.xhtml, user_edit.xhtml`, `user_new.xhtml, user_detail.xhtml` a řídící třída `UserBean`.

Během renderování klientské stránky `user_list` je v řídící třídě načten seznam uživatelů z databáze. Pro přístup k metodám nad databází slouží instance třídy `UserFacade` (Session bean).

Zobrazení seznamu uživatelů a přechod mezi zmíněnými klientskými stránkami pracuje na podobném principu jako v 6.2.4.

Při ukládání nového uživatele je provedena kontrola existence přihlašovacího jména jednoduchým dotazem nad databází. Kontrolu shody dvojice hesel ověřuje element `p:password`.

6.2.6 Log

Log reprezentuje klientská stránka `logs.xhtml` a řídící třída `LogBean`.

Během renderování klientské stránky je v řídící třídě načten seznam logů z databáze. Pro přístup k metodám nad databází slouží instance třídy `LogFacade` (Session bean).

Všechny metody řídících tříd, které provádějí CRUD operace nad databází obsahují navíc kód pro zaznamenání této události do logu.

6.2.7 Správa dokumentů

Správu dokumentů reprezentují klientské stránky `doc_list.xhtml, doc_edit.xhtml`, `doc_new.xhtml, doc_detail.xhtml` a řídící třída `DocBean`.

Během renderování klientské stránky `doc_list` je načten seznam dokumentů z databáze. Po přechodu na vybraný dokument (stránka `doc_detail`) je načten seznam kapitol a nálezů z databáze. Pro přístup k metodám nad databází zde slouží instance tříd `DocFacade, ChapterFacade, FindingFacade`. Řídící třída `DocBean` řídí pouze správu dokumentů, po přechodu například na stránku nálezu je řízení předáno odpovídající třídě.

Přechod mezi zmíněnými klientskými stránkami pracuje na podobném principu jako v 6.2.4.

6.2.8 Správa kapitol

Správu kapitol reprezentují klientské stránky `chapter_edit.xhtml, chapter_new.xhtml` a řídící třída `ChapterBean`.

Kapitolu je možné vytvářet, editovat pouze z klientské stránky detailu konkrétního dokumentu `doc_detail`. Pro přístup k metodám nad databází zde slouží instance třídy `ChapterFacade`.

Přechod mezi zmíněnými klientskými stránkami pracuje na podobném principu jako v 6.2.4.

6.2.9 Správa nálezů

Správu nálezů reprezentují klientské stránky `finding_detail.xhtml, finding_edit.xhtml`, `finding_new.xhtml` a řídící třída `FindingBean`.

Nález je možné vytvářet pouze z klientské stránky detailu konkrétního dokumentu `doc_detail`. Pro přístup k metodám nad databází zde slouží instance třídy `FindingFacade`.

Přechod mezi zmíněnými klientskými stránkami pracuje na podobném principu jako v 6.2.4.

6.2.10 Importování dokumentu

Importování dokumentu reprezentují klientské stránky `import.xhtml, import_wizard .xhtml` a řídící třídy `ImportBean a ImportWizard`.

Při renderování klientské stránky `import.xhtml` je ověřena existence auditu pomocí metody `auditsExists`, která využívá instanci třídy `AuditFacade` pro dotaz nad databází. V případě, že již doposud nebyl vytvořen žádný audit, je zobrazeno systémové hlášení popisující tento problém. V opačném případě je zobrazen formulář pro výběr a nahrání souboru na server. Nahrání souboru zajišťuje element `p:fileUpload`.

Po výběru souboru a stisknutí tlačítka `upload` je vyvolána metoda `upload(FileUploadEvent event)`, kde parametr event obsahuje informace o vytvořeném dočasném souboru na serveru. Tento soubor je poté překopírován metodou `copyFile` ze složky dočasných souborů do uživatelského adresáře. Po kliknutní na tlačítko `Průvodce`

importem je vyvolána metoda submit, která uloží seznam jmen nahraných souborů do objektu userSession reprezentující uživatelské sezení. Následuje přechod na klientskou stránku import_wizard.xhtml, která reprezentuje průvodce importem dokumentu.

Průvodce importem reprezentuje element p:wizard, ve kterém jsou vytvořeny tři kroky průvodce:

- Soubor – obsahuje informace o importovaném dokumentu.
- Struktura dokumentu – obsahuje formulář se strukturou zpracovávaného dokumentu.
- Rekapitulace.

Přechody mezi jednotlivými kroky řídí metoda onFlowProcess řídící třídy.

Při přechodu na druhý krok průvodce je vyvolána metoda parseDocument, která načte nastavení syn. analyzátoru z konfiguračního souboru a poté vytvoří nový objekt typu ImportWorker, který zpracuje aktuální dokument a jeho obsah uloží do objektu doc entitní třídy Doc. Po dokončení této náročné operace je zobrazen formulář, který obsahuje základní informace z titulní strany dokumentu a obsah (členění) dokumentu. Uživatel zde zkontroluje informace a v obsahu zaškrtne nálezy, které chce importovat.

Při přechodu na třetí krok průvodce je vyvolána metoda saveDocumentToDatabase, která uloží dokument se zvolenými nálezy do databáze. Každá část nálezu, která obsahuje obrázek (element img), je zpracována metodou parseImages, která přidá aktuální identifikátor vztahu (viz vztahy 3.1.6) v atributu src do seznamu vztahů a dále nastaví tomuto atributu cestu k obrázku na serveru. Dle seznamu vztahů jsou po uložení dokumentu do databáze překopírovány všechny potřebné obrázky z dokumentu do interní složky find-img.

6.2.11 Vyhledávání

Vyhledávání reprezentují klientské stránky search.xhtml, template.xhtml a řídící třída SearchBean.

Vyhledávací formulář je umístěn v šabloně stránek template a tudíž je přístupný z každé klientské stránky aplikace. Klasické vyhledávání provádí metoda search řídící třídy, která dále pomocí instance třídy FindingFacade volá metodu search pro vyhledání nálezů v databázi.

Rozšířené vyhledávání poskytuje sadu kritérií (viz tabulka 5.2) pro upřesnění výsledku vyhledávání. Objekt searchCr třídy SearchCriteria obsahuje vyhledávací kritéria. Rozšířené vyhledávání provádí metoda searchWithCriteria řídící třídy, která dále volá metodu extendSearch instance FindingFacade.

Seznam vyhledaných nálezů zobrazuje klientská stránka search.

6.2.12 Nákupní košík a generování dokumentu

Nákupní košík reprezentuje klientská stránka new_document.xhtml a průvodce generováním dokumentu stránka prepare_doc.xhtml. Obě stránky využívají řídící třídy UserLogin a GenerateBean.

Košík zobrazuje seznam nálezů uložených v objektu userSession, který obsahuje i metody pro vyprázdnění košíku. Při renderování této klientské stránky je volána metoda getCart, která vrací seznam nálezů.

Průvodce generováním dokumentu reprezentuje element p:wizard, ve kterém jsou vytvořeny čtyři kroky průvodce:

- Titulní strana – formulář pro vyplnění informací o dokumentu včetně počtu kapitol a jazyka.
- Kapitoly – formulář pro vyplnění názvů kapitol.
- Nálezy – formulář pro seřazení nálezů a kapitol.
- Rekapitulace – stručná rekapitulace a tlačítko pro vygenerování dokumentu.

Přechody mezi jednotlivými kroky řídí metoda `onFlowProcess` řídící třídy.

Stisknutím tlačítka `Generovat dokument` v posledním kroku průvodce dojde k vyvolání metody `actionGenerate`, která spustí generování. Po dokončení procesu dojde k přesměrování na klientskou stránku `download_list.xhtml`, kde je možné dokument stáhnout.

6.2.13 Přihlašování a uživatelské sezení

Přihlašování uživatele je provedeno z klientské stránky `login.xhtml`, odkud je volána metoda `login` řídící třídy `UserLogin`. Tato metoda pomocí instance třídy `UserFacade` ověří existenci uživatele v databázi.

V případě chybných přihlašovacích údajů je inkrementován čítač `badAccessCnt` a pokud dosáhne hodnoty tři, je další povinnou položkou pro přihlášení pole Captcha.

Po úspěšném přihlášení uživatele je vytvořen objekt uživatelského sezení `userSession` třídy `UserSession`, jehož inicializace spočívá v:

- načetní nastavení uploadu do objektu `fileUpSet` typu `FileUploadSettings`,
- vytvoření uživatelského adresáře,
- nastavení cesty k uživatelskému adresáři.

6.3 Syntaktický analyzátor

Syntaktický analyzátor reprezentuje třída `ImportWorker`, která obsahuje metody pro zpracování dokumentu docx. V konstruktoru třídy je provedena inicializace objektů `xmlset, userDir a docxFilePath`, kde:

- `xmlset` reprezentuje nastavení syntaktického analyzátoru,
- `userDir` obsahuje absolutní cestu k uživatelskému adresáři a
- `docxFilePath` obsahuje absolutní cestu ke zpracovávanému souboru.

Veřejná metoda `startParseDocument` provede zpracování dokumentu docx, které probíhá v těchto krocích:

1. Dearchivace dokumentu docx.
2. Nastavení absolutní cesty k adresáři obsahující dearchivovaný dokument.
3. Zpracování titulní strany dokumentu.
4. Transformace dokumentu docx na dokument XHTML.
5. Načtení vztahů dílčích součástí dokumentu.

6. Příprava nového strukturovaného dokumentu.
7. Zpracování obsahu dokumentu XHTML.
8. Vytvoření obsahu (členění) dokumentu.
9. Vytvoření kapitol s nálezy.

Tato sekce popisuje zmíněný postup zpracování dokumentu.

6.3.1 Dearchivace dokumentu

Dearchivaci dokumentu provádí metoda `decompressDocx`, která vytvoří kopii stávajícího dokumentu a vloží na konec názvu příponu zip. Poté provede dearchivaci vytvořené kopie, kterou po dokončení této operace smaže.

6.3.2 Zpracování titulní strany

Zpracování titulní strany provádí metoda `parseFrontPage`, která vytvoří objekt `doc`, reprezentující xml model souboru `word/document.xml`, který se nachází v rozbaleném adresáři dokumentu.

Z objektu doc je vybrán seznam elementů `w:txbxContent`, které reprezentují jednotlivá textová pole, ze kterých budou získána potřebná data. Velikost seznamu musí být větší nebo rovna pozicím textových polí daných v objektu `xmlset`. V případě nesplnění této podmínky je vyvolána výjimka upozorňující na tuto chybu.

Poté jsou ze seznamu vybrány dva elementy dle pozice dané v `xmlset`, první reprezentuje textové pole s údaji název – společnost a druhý datum – autoři.

Z obou elementů jsou vybrány seznamy vnitřních elementů `w:p` reprezentující odstavce uvnitř textového pole. Velikost obou seznamů musí být větší nebo rovna pozicím odstavců daných v objektu `xmlset`. V případě nesplnění této podmínky je vyvolána výjimka upozorňující na tuto chybu.

Nakonec jsou z jednotlivých odstavců dle daných pozic získána potřebná data a poté vytvořen nový objekt `doc` entitní třídy `Doc` s nastavenými vlastnostmi.

6.3.3 Transformace dokumentu docx na dokument XHTML

Transformaci dokumentu provádí metoda `convertDocumentToHTML`, která dále využívá metod knihovny Javadocx. Kód 6.3 zobrazuje transformaci dokumentu. Převedený dokument je nakonec uložen do souboru.

```
TransformDoc transDoc = new TransformDoc();
transDoc.setStrFile(docxFilePath);
transDoc.generateXHTML();
transDoc.validatorXHTML();
```

Kód 6.3: Transformace dokumentu docx na dokument XHTML.

6.3.4 Načtení vztahů dílčích součástí dokumentu

Vztahy součástí dokumentu (viz 3.1.6) jsou důležité pro práci s obrázky během zpracování XHTML dokumentu. Jedná se o obrázky reprezentující `skill` (počet hvězdiček nálezu). Jednotlivé vztahy jsou načteny ze souboru `word/_rels/document.xml.rels` a dále namapovány do objektu `Map<String, String>`, kde klíčem je `Id` a hodnotou `Target`.

Elementy `img` uvnitř transformovaného XHTML dokumentu obsahuji v atributu `src` identifikátor vztahu a nyní je možné pomocí namapovaných vztahů získat cestu.

6.3.5 Příprava strukturovaného dokumentu

Příprava strukturovaného dokumentu XHTML spočívá ve vytvoření objektového modelu dokumentu z řetězce, který obsahuje základní strukturu elementů `html, head, body`. Řetězec je zpracován pomocí metody `Parser.parse` knihovny Jsoup.

6.3.6 Zpracování obsahu dokumentu XHTML

Zpracování obsahu dokumentu XHTML provádí metoda `parseContent`, která nejprve načte XHTML dokument jako objektový model a poté provede výběr všech elementů obsahující atribut `class` s hodnotou h1 (výběr kapitol). Následně je zkontrolována existence struktury dokumentu a nadpisů. Pokud se v dokumentu nevyskytují některé potřebné elementy, je vyvolána příslušná výjimka charakterizující daný problém.

Poté je proveden průchod dokumentem po jednotlivých kapitolách a dále jsou zpracovávány veškeré elementy významově spojené s touto kapitolou. Každý element obsahující atribut `class` s hodnotou některého stylu z konfigurace analyzátoru je zpracován jako nadpis.

Pro každý nadpis, mimo nadpis nálezu, je vytvořen reprezentující XHTML zápis, který obsahuje pouze styl nadpisu a název. Tyto zápisy jsou převedeny na objektový model a vkládány do strukturovaného dokumentu.

Při zpracování nadpisu nálezu je provedena analýza rizika, skill a obsahu, který je reprezentován elementy nacházejícími se ihned za elementem nadpisu. Nový nadpis nálezů s obsahem je zobrazen v XHTML kódu 5.8. Analýza typu rizika je provedena na základě konfigurace analyzátoru. Pro analýzu skill jsou využity dílčí vztahy dokumentu, kde je na základě identifikátoru vztahu uvnitř elementu zjištěn vztah a následná cesta k obrázku reprezentující skill. Poté je provedeno porovnání referenčního a analyzovaného obrázku, ze kterého je zjištěna hodnota skill.

Zjednodušený formát strukturovaného dokumenu zobrazuje kód 5.8.

6.3.7 Vytvoření členění dokumentu

Členění dokumentu je vytvořeno v metodě `parseSummary`, kde je proveden průchod strukturovaného dokumentu po jednotlivých kapitolách. Pro každý nadpis reprezentující danou sekci je vytvořen objekt typu `Header`, který obsahuje následující části:

- nadpis kapitoly s číslováním úrovně,
- příznak nadpisu nálezu,
- objekt reprezentující zvolený audit dané kapitoly,
- příznak kapitoly obsahující nálezy,

- příznak importování nálezu do databáze.

Seznam těchto vytvořených objektů je poté zobrazen v uživatelském prostředí jako členění dokumentu s možností volby auditu dané kapitoly a výběru nálezů, které budou importovány do databáze. Příznak kapitoly obsahující nálezy slouží pro renderování komponenty pro výběr auditu.

6.3.8 Vytvoření kapitol s nálezy

Vytvoření objektů reprezentujících kapitoly provádí metoda `getChapters`. Průchodem strukturovaného dokumentu jsou postupně vytvářeny objekty kapitol typu `Chapter`, které jsou přidávány do seznamu všech kapitol. Každé kapitole obsahující nálezy je přiřazen seznam nálezů typu `Finding`. Odkaz na vytvořený seznam kapitol je vložen do objektu dokumentu vytvořeného pří zpracování titulní strany viz 6.3.2.

6.4 Generátor dokumentu

Generátor dokumentu reprezentuje třída `GenerateDoc`, která obsahuje metody pro vytvoření nového dokumentu a metody z knihovny Docx4j. V konstruktoru třídy je provedena počáteční inicializace následujících objektů:

- `doc` – reprezentuje nový dokument
- `docParts` – seřazený seznam kapitol a nálezů,
- `userDirPath` – absolutní cesta k uživatelskému adresáři,
- `realPath` – absolutní cesta ke kořenovému adresáři aplikace,
- `fileName` – název generovaného souboru.

Metoda `startGenerate` spouští proces generování nového dokumentu, které probíhá v následujících krocích:

1. Příprava a načtení cílového souboru.
2. Nastavení titulní strany dokumentu.
3. Vložení kapitol a nálezů.

6.4.1 Příprava a načtení cílového souboru

Na základě jazyka, ve kterém je dokument napsán, je vytvořena kopie šablony, která je uložena do uživatelského adresáře `userDirPath`. Tato šablona slouží pro doplnění kapitol a nálezů. Cílový soubor je poté načten jako objektový model pomocí knihovny Docx4j.

6.4.2 Nastavení titulní strany dokumentu

Na titulní straně šablony jsou umístěny speciální řetězce, které budou nahrazeny. Jedná se o doplnění názvu dokumentu, data vytvoření a autoři.

Operaci nahrazení řetězce na titulní straně dokumentu provádí metoda, která provede zanoření v objektovém modelu dokumentu až na elementy `CTTextbox`, které reprezentují

jednotlivá textová pole. Uvnitř textových polí se dále vyskytují elementy `w:p` reprezentující jednotlivé odstavce. Obsah odstavce, který obsahuje hledaný řetězec je nahrazen vkládanými daty.

6.4.3 Vložení kapitol a nálezů

Vložení kapitol a nálezů provádí metoda `addDataToDocx`, která prochází seznam objektů `docParts` a vytváří objektový model jednotlivých kapitol a nálezů v načteném cílovém docx dokumentu.

Nadpis kapitoly dokumentu reprezentuje nově vytvořený objekt, který reprezentuje odstavec `w:p`. Tomuto objektu je nastaven styl a název kapitoly. Vytvořený objekt je přidán do těla dokumentu.

Nadpis nálezu dokumentu je vytvořen na stejném principu jako nadpis kapitoly, s tím rozdílem, že za nadpis nálezu je navíc vložen obrázek značící riziko. Pomocí knihovny Docx4j je vytvořen element obsahující obrázek odpovídající hodnotě rizika. Cesta k obrázku je určena pomocí `realPath`. Vytvořený objekt je přidán do těla dokumentu.

Tabulku, obsahující popisný název rizika a hodnotu skill, reprezentuje nově vytvořený objekt, kterému je dále přidáno nastavení, struktura sloupců a obsah jednotlivých buněk. Hodnotu skill reprezentuje trojice obrázků, která je do buňky vložena stejně jako při vytvoření obrázku v nadpisu nálezu. Vytvořený objekt je přidán do těla dokumentu.

Nadpis části nálezu je vytvořen na stejném principu jako vytvořen nadpis kapitoly. Obsah části nálezů je vytvořen pomocí funkce transformace XHTML na objektový model docx, kterou poskytuje knihovna Docx4j. Obrázky vyskytující se uvnitř kódu jsou také zpracovány.

6.5 Filtrování protokolu HTTP

Pro filtrování protokolu HTTP byly implementovány Servlet filtry. Jedná se o speciální třídy, které upravují HTTP požadavky na základě seřazení filtrů v konfiguračním souboru `web.xml`.

Tato sekce popisuje implementaci HTTP filtru pro kódování znaků a zabezpečovací filtr.

6.5.1 Kódování znaků

Filtr pro kódování znaků reprezentuje třída `CharEncodingFilter`, která implementuje rozhraní `Filter`. V našem případě byla provedena implementace metody `doFilter`, která nastaví příchozím i odchozím požadavkům kódování znakové sady UTF-8.

6.5.2 Zabezpečovací filtr

Filtr pro zabezpečení obsahu reprezentuje třída `LoggedFilter` a pracuje na stejném principu jako filtr pro kódování znaků (viz sekce 6.5.1). Metoda `doFilter` zkontroluje stav sezení přihlášeného uživatele. V případě, že uživatel není přihlášen, je provedeno přesměrování na přihlašovací stránku. Pokud má uživatel omezená práva přístupu na některé zabezpečené stránky, je přesměrován na stránku zobrazující status „Přístup zamítnut“.

Kapitola 7

Testování

Tato kapitola popisuje testování funkčnosti, použitelnosti, aplikačních serverů, kompatibility a bezpečnosti aplikace. U každého testu je popsán cíl testu, který pojednává o získaných znalostech. Poté je popsán způsob provedení testu a jeho průběh. Na závěr je provedeno zhodnocení dosažených výsledků.

7.1 Funkčnost

Tato sekce popisuje testování funkčnosti odkazů, kde bude ověřena správnost navigačních pravidel a přechodů mezi jednotlivými stránkami. Dále je popsáno testování formulářů na nepovolené vstupy a nakonec testování syntaktické analyzátoru.

7.1.1 Kontrola odkazů

Cílem testování kontroly odkazů bylo ověření funkčnosti přechodu mezi jednotlivými stránkami aplikace, které definují navigační pravidla (viz sekce 5.2.5).

V aplikaci byly postupně prováděny přechody mezi jednotlivými stránkami. Z každé stránky aplikace byly testovány odkazy hlavního navigačního menu v levé části aplikace a vnitřní odkazy stránky. U chybných odkazů byla upravena navigační pravidla.

Po dokončení testování nebyly nalezeny chybné odkazy. V případě vyššího počtu odkazů by bylo vhodné testovat aplikaci například pomocí nástroje Xenu.

7.1.2 Kontrola formulářů

Cílem testování kontroly formulářů bylo ověření nepovolených vstupů. Validace formuláře musí být spolehlivá.

V aplikaci byly postupně testovány všechny vyskytující se formuláře (Správa dokumentů, Správa nálezů, Správa uživatelů a další). Do každého pole formuláře byly zadány různé kombinace vstupů a bylo sledováno chování aplikace.

Testem bylo prokázáno, že všechny formuláře aplikace ošetřují nepovolené vstupy. Při zadání chybného vstupu bylo vždy zobrazeno chybové hlášení, které upozorňuje uživatele na rozsah povolených hodnot.

7.1.3 Syntaktický analyzátor

Cílem testování syntaktického analyzátoru je ověřit funkčnost zpracování docx dokumentů a specifikovat jeho kolizní stavy.

Pro testování byla vytvořena sada testovacích dokumentů obsahující:

- Sada A – Dokumenty různých velikostí, které splňují specifikaci.
- Sada B – Dokumenty s drobnými odchylkami od specifikace (špatný formát tabulek, nadpisů a podobně).
- Sada C – Dokumenty bez vztahu k aplikaci (mimo specifikaci).

Pro každý dokument byl proveden pokus o import dat do databáze a přitom bylo zaznamenáno chování aplikace. Výsledky testů ukazuje tabulka 7.1.

Typ	**Počet stran**	**Velikost**	**Doba zpracování**	**Chování aplikace**
Dokument A	100	643 kB	15,9 s	Import OK
Dokument A	200	1 MB	31,6 s	Import OK
Dokument A	500	1,6 MB	1 min 20 s	Import OK
Dokument A	1000	3,2 MB	2 min 54 s	Import OK
Dokument B	34	470 kB	8 s	Import OK, chybně formátované nálezy byly přeskočeny.
Dokument C	15	118 kB	-	Hlášení: Chybné nastavení titulní strany.

Tabulka 7.1: Výsledky testování syn. analyzátoru.

Při testování dokumentů aplikace reagovala podle očekávání. Dokumenty s korektním formátováním byly zpracovány a importovány. Dokumenty, které obsahovaly některé chyby ve formátování, byly importovány a chybné části přeskočeny. Dokumenty bez vztahu k aplikaci nebylo možné zpracovat a aplikace vypsala hlášení o chybném nastavení titulní strany dokumentu.

7.2 Použitelnost

Cílem testovaní použitelnosti aplikace bylo ověřit jednoduchost a intuitivnost při práci s aplikací.

Pro testování aplikace bylo vytvořeno deset uživatelských účtů pro deset testerů, kteří byli seznámeni se základní funkčností aplikace. Každému testeru byl přiřazen testovací soubor obsahující nastavení syn. analyzátoru a seznam následujících úkolů:

- přihlášení do aplikace,
- nastavení syn. analyzátoru dle zadání,
- vytvoření dvou auditů,
- změnu osobních údajů,
- importování testovacího dokumentu,
- vyhledání nálezů s typem rizika nízká, které obsahují slovo xss,
- pět nálezů ze seznamu vyhledaných vložit do košíku,

- nálezy v košíku vygenerovat do nového dokumentu a tento dokument stáhnout.

Kontrola byla prováděna pozorováním testerů při plnění úkolu. Většina testerů byla schopna aplikaci ovládat a splnit zadaný úkol. Ve třech případech se naskytly problémy s nastavením pozic textových polí v nastavení konfigurace syntaktického analyzátoru. Po poskytnutí nápovědy byly překonány i tyto problémy. Z výsledku testu lze konstatovat, že uživatelské prostředí je srozumitelné a snadno ovladatelné.

7.3 Aplikační servery

Aplikace byla po celý vývoj testována na aplikačním serveru GlassFish. Všechny chyby, které se vyskytly, či byly objeveny v průběhu vývoje, byly opravovány.

Cílem tohoto testovaní bylo ověřit běh konečné aplikace na aplikačním serveru TomEE a objevit tak další doposud neznámé chyby, které mohou způsobit problémy se spuštěním aplikace.

V nastavení aplikace byl ve vývojovém prostředí Netbeans přiřazen aplikační server TomEE s nakonfigurovanou EJB vrstvou (spojení s databází). Dále test probíhal v jednotlivých iteracích, kdy aplikace byla přeložena, spuštěna a následně byly opraveny odhalené chyby.

V průběhu testování bylo provedeno několik iterací, ve kterých byly objeveny problémy se spuštěním aplikace. Komponenty technologie JSF obsahující atribut `binding`, byly chybně zobrazovány a po několika kliknutí byla aplikace zcela nefunkční. Atribut byl odstraněn a jeho význam nahrazen jiným principem. Na serveru GlassFish se tento problém nikterak neprojevoval.

Výsledkem testování je fungující aplikace na serveru TomEE se stejným chováním jako na serveru GlassFish.

7.4 Kompatibilita

Cílem testování kompatibility je ověřit zobrazení a práci s aplikací v nejnovějších verzích webových prohlížečích Firefox (verze 21), Chrome (verze 26), Opera (verze 12), Internet Explorer (verze 9).

V průběhu testu byla aplikace spuštěna ve všech výše uvedených prohlížečích a na základě pozorování byly zaznamenány různé odlišnosti v zobrazení vzhledu a komponent aplikace.

Test prokázal kompatibilitu z pohledu zobrazení vzhledu a komponent ve webových prohlížečích Firefox, Opera a Chrome. V prohlížeči Internet Explorer byl zjištěn problém se zobrazením některých obrázků a pomocí úpravy kaskádových stylů vyřešen.

7.5 Zabezpečení

Cílem testování zabezpečení aplikace je ověřit přístupnost zabezpečeného obsahu z vnějšího prostředí. Jedná se o přístup k obsahu aplikace bez řádného přihlášení uživatele a po přihlášení uživatele s omezenými právy.

Do adresní řádky webového prohlížeče jsou postupně zadávány URL adresy všech zabezpečených stránek a je sledováno chování aplikace. Poté je provedeno přihlášení uživatele s omezenými právy a opět jsou zadávány URL adresy zabezpečených stránek. Poté

je proveden test přístupnosti k nahraným a vygenerovaným dokumentům (obdobně jako u zabezpečených stránek).

Při pokusu o přístup na všechny testované zabezpečené stránky aplikace reagovala přesměrováním na stránku s přihlášením. Při pokusu o přístup všech testovaných stránek z neadministrátorských účtů aplikace reagovala zamítnutím. Při pokusu o přístup k zabezpečeným souborů aplikace vyžadovala autorizaci. Všechny získané výsledky splňují očekávané předpoklady.

Z hlediska útoku známého pod názvem SQLInjection lze s jistotou tvrdit, že je aplikace bezpečná. Java Persistence API zaručuje tvorbu bezpečných SQL dotazů a tedy nelze z formulářů napadnout databázi.

Kapitola 8

Závěr

Cílem této diplomové práce bylo nastudovat problematiku webových aplikací implementovaných v prostředí Java Enterprise Edition a možnosti zpracování dokumentů formátu docx. Poté navrhnout aplikaci, která bude umožňovat import, uchovávání, vyhledávání, editaci a tvorbu nových dokumentů formátu docx. Zpracování dokumentu bude provádět konfigurovatelný syntaktický analyzátor. Dále tuto aplikaci implementovat a otestovat z pohledu funkčnosti a bezpečnosti.

Všechny body zadání byly splněny a vytvořená aplikace pro správu dokumentů je plně funkční. Nastavení syntaktického analyzátoru umožňuje volit formát zpracovávaného souboru. Pomocí průvodce importování dokumentu lze zpracovat soubory s požadovaným formátováním, přičemž před uložením dat do databáze je zobrazen obsah dokumentu. Ze zobrazené struktury dokumentu může uživatel zvolit částí, které budou uloženy do databáze. Z dat uložených v databázi lze generovat nové dokumenty.

Bylo provedeno testování z pohledu funkčnosti, použitelnosti, kompatibility a bezpečnosti. Testování použitelnosti ověřilo jednoduchost ovládání aplikace na několika uživatelích, kteří byli schopni s uživatelským prostředím pracovat a plnit zadané úkoly. Aplikace je kompatibilní ve čtyřech nejrozšířenějších webových prohlížečích, je zabezpečena proti neautorizovanému přístupu k datům a odolná vůči základním typům útoků (SQLInjection). Syntaktický analyzátor je schopen zpracovat dokumenty s auditními daty. Získané výsledky odpovídají předpokládanému chování aplikace a podrobněji jsou popsány v kapitole 7 zabývající se testováním.

Pro vypracování této práce bylo potřeba nastudovat implementaci webových aplikací v prostředí Java Enterprise Edition a dostupné frameworky pro prezentační vrstvu aplikace a práci s databází. Následovalo studium specifikace standardu Open XML definujícího formát docx. Dále bylo nutné provést specifikaci požadavků pro upřesnění funkčnosti aplikace a provést její návrh. Podle vytvořeného návrhu implementovat a otestovat aplikaci.

8.1 Rozšíření aplikace

V této sekci následuje seznam možných rozšíření aplikace, které nebyly v rámci diplomové práce z časových a jiných důvodů implementovány. Jedná se o tato rozšíření:

- Vícejazyčnost uživatelského prostředí.
- Rozšířené nastavení syntaktického analyzátoru v průvodci importování dokumentu.

- Implementace analýzy chybné struktury importovaného dokumentu (což uživateli usnadní nastavení syn. analyzátoru).
- Implementace přímého generování dokumentů uložených v databázi a jejich export do PDF.
- Implementace správy uživatelských souborů.
- Pokročilejší formát generovaných dokumentů.

Literatura

[1] Ecma International: Standard ECMA-376, Office Open XML File Formats [online]. Dostupné z `http://www.ecma-international.org/publications/standards/Ecma-376.htm`, 2012-12 [cit. 2012-12-15].

[2] Goncalves, A.: *Beginning Java EE 6 Platform with GlassFish 3*. Springer Science New York, 2010, ISBN 978-1-4302-2889-9, 537 s.

[3] ICEsoft Technologies Inc.: Ice Faces library [online]. Dostupné z `http://www.icesoft.org/java/home.jsf`, 2013 [cit. 2013-01-05].

[4] J. Zendulka, I. Rudolfová: *Databázové systémy, studijní opora*. FIT VUT v Brně, 2006, 217 s.

[5] Oracle: The Java EE 6 Tutorial [online]. `http://docs.oracle.com/javaee/6/tutorial/doc/docinfo.html`, 2012-07 [cit. 2012-12-01].

[6] Oracle: GlassFish [online]. Dostupné z `http://glassfish.java.net`, 2012 [cit. 2012-12-28].

[7] Plutext Pty Ltd: Docx4j library [online]. Dostupné z `http://www.docx4java.org`, 2012 [cit. 2012-12-18].

[8] PrimeFaces: Prime Faces library [online]. Dostupné z `http://www.primefaces.org`, 2013 [cit. 2013-01-05].

[9] Red Hat JBoss: Hibernate [online]. Dostupné z `http://www.hibernate.org/`, 2012 [cit. 2013-01-08].

[10] Schildt, H.: *Java 7 Výukový kurz*. Computer Press, 2012, ISBN 978-80-251-3748-2, 664 s.

[11] The Apache Software Foundation: Apache TomEE [online]. Dostupné z `http://tomee.apache.org`, 2012 [cit. 2012-12-28].

[12] Turrion, M.: Javadocx library [online]. Dostupné z `http://www.javadocx.com`, 2012 [cit. 2012-12-26].

Příloha A

Obsah DVD

Přiložené DVD obsahuje projekt vytvořený v nástroji Netbeans se zdrojovými soubory, knihovny, potřebné pro chod aplikace, konfigurační soubory, šablony docx dokumentů pro generování, technickou zprávu ve formátu pdf a zdrojové soubory technické zprávy napsané v programu LaTeX.

A.1 Adresářová struktura

`Aplikace/`

- `lib/` – použité knihovny.
- `nbproject/` – konfigurační soubory projektu v Netbeans.
- `setup/` – konfigurační soubory pro nastavení aplikačního serveru.
- `src/` – zdrojové soubory v jazyce Java.
- `web/` – konfigurační soubory a zdrojové soubory prezentační vrstvy.
- `database.sql` – SQL skript obsahující schéma databáze s inicializačními
- `README` – Soubor obsahující přihlašovací údaje s daty.

`Technická zpráva/`

- `latex/` – zdrojové soubory a obrázky pro překlad technické zprávy.
- `zprava.pdf`

Příloha B

Stručný návod pro spuštění aplikace

Tato příloha obsahuje stručný návod pro spuštění projektu v nástroji Netbeans. Je potřeba nastavit některé konfigurační soubory prezentační vrstvy. Přihlašovací údaje do aplikace lze nalézt v souboru `README`.

Doporučený postup pro spuštění aplikace:

1. V databázovém prostředí mysql (phpmyadmin) provést vytvoření nové databáze s porovnáním `utf8_general_ci` a dále importovat soubor `database.sql`, který vytvoří tabulky databáze.

2. Otevřít projekt ze složky `Aplikace` v nástroji Netbeans. V nastavení projektu v sekci run přiřadit aplikační server GlassFish a v sekci libraries přiřadit všechny knihovny ze složky `lib`.

3. V konfiguračním souboru `web/WEB-INF/web.xml` nastavit absolutní cestu k adresáři dočasných souborů `temp` (element `param-value` u filtru `PrimeFaces FileUpload Filter`).

4. V konfiguračních souborech `src/conf/persistence.xml` a `setup/glassfish-sources.xml` nastavit spojení s databází. V souboru `persistence.xml` navíc nastavit absolutní cestu k adresáři pro uložení indexu vyhledávání.

5. Překlad a spuštění projektu.